

Using Python to Find/Replace with Regular Expressions from T-SQL

Erland Sommarskog
Data Platform MVP
esquel@sommarskog.se

Slides and scripts: **<http://www.sommarskog.se/present>**

Warning

If you don't know regular expressions, you
will not learn them in ten minutes.

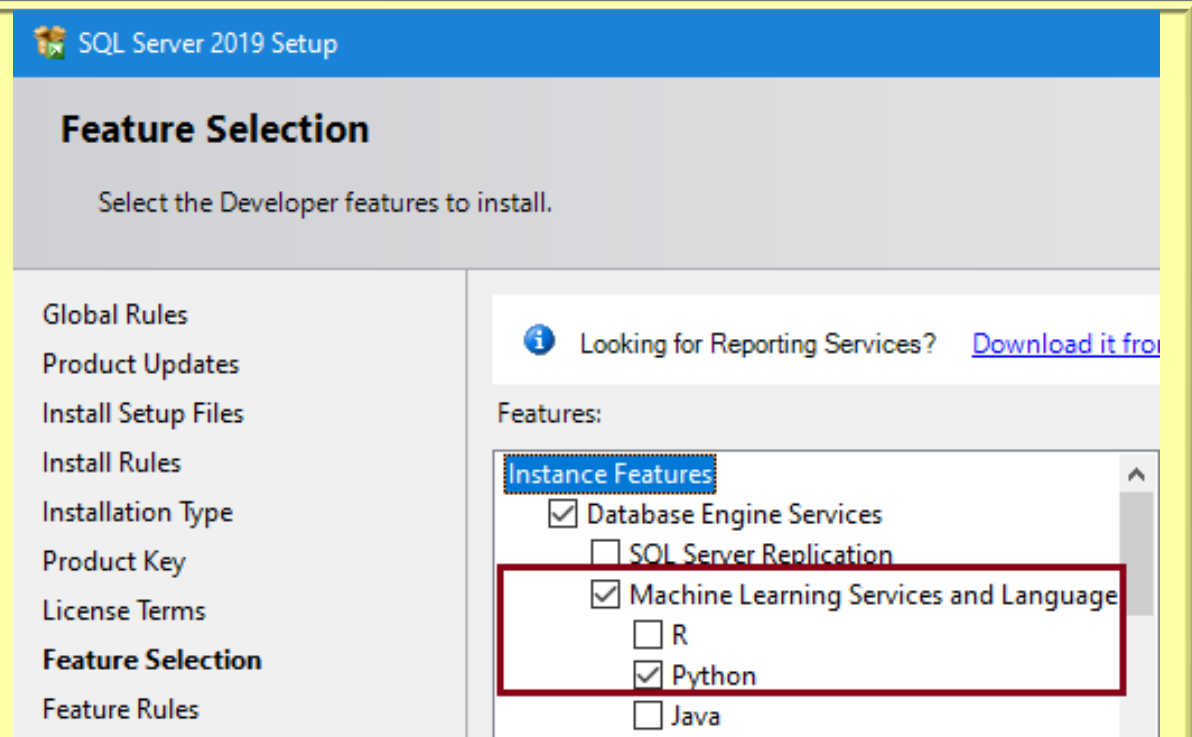
... but you may get a glimpse of their
powers.

We Want Regular Expressions!

- LIKE is very limited. Replace() only works with fixed strings.
- CLR has the RegEx class. A lot of red tape, unless you're already using it.
- The Machine Learning Services since SQL 2016:
 - R, since SQL 2016.
 - Python, since SQL 2017.
 - Java, since SQL 2019.
 - (And since SQL 2019, you can add your own language.)

Preconditions

1. Python support must be installed.
2. Server configuration parameter **external scripts enabled** must be 1. (Default 0.)
3. User must have the database permission **EXECUTE ANY EXTERNAL SCRIPT**.



How Scripts Are Executed

- Python scripts are started by `sp_execute_external_script`.
- The script is executed by the *Launchpad* process.
- It is separate from the SQL Server process.
- By default, the Launchpad runs with very limited permissions in the OS.

Example: Clean up CSVs

We Have	We want
1,2,3,4	1, 2, 3, 4
Adam, Betty, Charlie	Adam, Betty, Charlie
XV, LXI,VII,MCM	XV, LXI, VII, MCM
This, is, perfect	This, is, perfect
No comma in this text	No comma in this text
ABC,,DDD,, F,, XYZ	ABC,, DDD,, F,, XYZ

[Examples.sql](#)

Heart of the Matter

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
,  
  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script
```

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
,  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script
```

Defines the language
of the script.


```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
,  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script
```

The data set the Python script is to work on.

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
,  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script
```

Name of the Python variable
that receives the input data.

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
,  
  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script
```

Name of the Python variable
that holds the output data.

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
,  
  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script
```

The script itself

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
,  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',
```

Import of Python packages.

re = Regular expressions.

pandas = SQL Server sends and receives data to/from Python over pandas "data frames".

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r"\s*(\w)", r"\1", str)  
    for str in Data["txt"]])  
'  
  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script
```

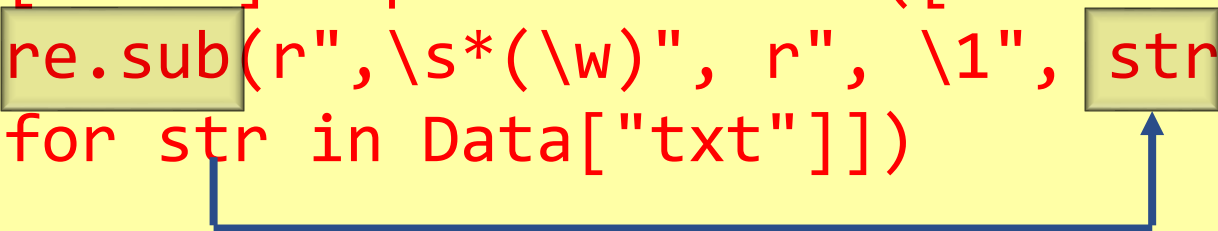
The diagram illustrates the data flow in the provided T-SQL code. It features three blue arrows: one originates from the 'txt' column in the SQL query 'SELECT id, txt FROM Playdata', points to the 'Data["txt"]' variable in the Python script, and then continues to the 'for str in Data["txt"]' loop; a second arrow points from the 'Data["txt"]' variable to the 'pandas.Series' constructor; and a third arrow points from the 're.sub' function call to the 'Data["txt"]' variable, indicating that the function processes the data from the 'txt' column.

“Python/Pandas mumbo-jumbo”

```
DECLARE @python_script nvarchar(MAX) = N'
```

```
import re, pandas
```

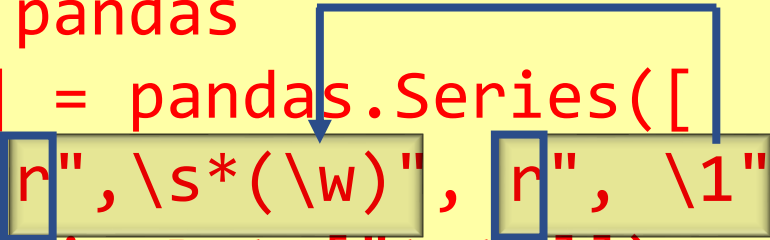
```
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
,
```

A diagram consisting of two grey rectangular boxes. The first box is positioned over the variable 'str' in the line 'for str in Data["txt"]'. The second box is positioned over the variable 'str' in the line 're.sub(r",\s*(\w)", r", \1", str)'. A blue line starts from the bottom of the first box, goes down, then right, and finally up to the bottom of the second box, indicating the flow of the variable 'str' from its definition to its use in the 're.sub' function.

```
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script
```

re.sub is the method that performs the find/replace. It operates on the Python variable **str**, defined on the line below.

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["txt"] = pandas.Series([  
    re.sub(r",\s*(\w)", r", \1", str)  
    for str in Data["txt"]])  
'  
  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, txt FROM Playdata'
```



- **, \s*(\w)** What to replace: Comma, followed by zero or more white space and one alphanumeric character.
- **, \1** What to insert: Comma, single space, **\1** maps back to the parentheses, that is that word character.
- Observe the **r** before the quotes – raw string to protect the backslashes.

Example: Remove Titles

We Have	We want
Big Business Inc.	Big Business
Mary Jones Sr.	Mary Jones
English Drinks Ltd	English Drinks
Frank Hinc	Frank Hinc
Dr. Michael Keen	Michael Keen
Mr. John King Sr.	John King
Titles: inc, ltd, Mr, Dr, Sr	

[Examples.sql](#)

Building RegExp in T-SQL

```
SELECT @RegExp =  
    string_agg('(\s*\b' + title + '\b\.*\s*)', '|')  
FROM Titles
```

(\s*\bDr\b\.*\s*)|(\s*\binc\b\.*\s*)|(\s*\bltd\b\.*\s*)|(...

\s* = Zero or more white spaces.

\b = Word boundary

\.* = Zero or more period characters.

Dr, inc, etc. = The actual titles.

| = Alternate. Any of the patterns in parentheses match.

DECLARE @python_script **Parameter list and parameters to Python script. Must come at end!**

```
import re, pandas
Data["cleannname"] = pandas.Series([
    re.sub(RegExp, r"", str, flags=re.IGNORECASE)
    for str in Data["fullname"]])
'
```

```
EXEC sp_execute_external_script @language = N'Python',
    @input_data_1 = N'SELECT id, fullname FROM Names',
    @input_data_1_name = N'Data',
    @output_data_1_name = N'Data',
    @script = @python_script,
    @params = N'@RegExp nvarchar(MAX)',
    @RegExp = @RegExp
```

```
DECLARE @python_script nvarchar(MAX) = N'  
import re, pandas  
Data["cleannname"] = pandas.Series([  
    re.sub(RegExp, r"", str, flags=re.IGNORECASE)  
    for str in Data["fullname"]])  
'  
  
EXEC sp_execute_external_script @language = N'Python',  
    @input_data_1 = N'SELECT id, fullname FROM Names',  
    @input_data_1_name = N'Data',  
    @output_data_1_name = N'Data',  
    @script = @python_script,  
    @params = N'@RegExp nvarchar(MAX)',  
    @RegExp = @RegExp
```

By default, re.sub works case-sensitive, so need to set flag.

```

DECLARE @python_script nvarchar(MAX) = N'
import re, pandas
Data["cleanname"] = pandas.Series([
    re.sub(RegExp, r"", str, flags=re.IGNORECASE)
    for str in Data["fullname"]])
'

EXEC sp_execute_external_script @language = N'Python',
    @input_data_1 = N'SELECT id, fullname FROM Names',
    @input_data_1_name = N'Data',
    @output_data_1_name = N'Data',
    @script = @python_script,
    @params = N'@RegExp nvarchar(MAX)',
    @RegExp = @R

```

Adding one more column to the result set by introducing a new name.

- Erland Sommarskog: esquel@sommarskog.se.
- Slides and script:
<http://www.sommarskog.se/present>.
- SQL Docs: [*Quickstart: Run simple Python scripts with SQL machine learning*](#) and [*Quickstart: Data structures and objects using Python with SQL machine learning*](#).
- Python: [*re — Regular expression operations*](#).
- (Pandas: [*Intro to data structures*](#).)