

Cómo usar Python en SQL Server 2017 para obtener análisis avanzado de datos

May 28, 2018 by [Prashanth Jayaram](#)

El 19 de Abril de 2017, Microsoft tuvo una conferencia en línea llamada *Microsoft Data Amp* para mostrar cómo las últimas innovaciones de Microsoft ponen los datos, el análisis y la inteligencia artificial en el corazón de la transformación del negocio. Microsoft, a través de los últimos años, ha hecho grandes avances en acelerar el ritmo de innovación para permitir al negocio cumplir con las demandas de un mercado dinámico y aprovechar el increíble poder de los datos, de manera más segura y rápida que nunca antes.

Después de la conferencia, había algunas preguntas que algunos de nosotros teníamos aún. ¿Está Microsoft SQL Server 2017 emergiendo como una solución empresarial para ciencia de datos? ¿Provee las capacidades requeridas para que el motor sea capaz de manejar datos enormes? Parece que la respuesta es "Sí", ya que, comenzando con el lanzamiento de CTP 2.0 de SQL Server 2017, Microsoft ha traído inteligencia de datos basada en Python en SQL Server.

Python ha obtenido mucho interés recientemente como el lenguaje de elección para análisis de datos. Este lenguaje tiene el conjunto de librerías correcto para *análisis de datos* y *modelado predictivo*, sin mencionar una curva de aprendizaje menos pronunciada.

Las tendencias crecientes de ciencia y modelado de datos predicen un crecimiento masivo en los datos en los años por venir. La propulsión hacia la innovación y adaptación que liderarán tendencias en tecnología de datos puede intrigarnos lo suficiente para hacernos dar un vistazo al lanzamiento actual de SQL Server 2017.

La ciencia de Datos es una combinación de Minería de Datos, Aprendizaje de Máquina, Análisis y Grandes Datos. La integración de SQL 2016 con un lenguaje de ciencia de datos, R, al motor de la base de datos provee una interfaz que puede ejecutar eficientemente modelos y generar predicciones usando los servicios SQL R. Python construye sobre la base propuesta por los Servicios R en SQL Server 2016, y extiende ese mecanismo para incluir soporte para Python para análisis en la base de datos y aprendizaje de máquina.

Por otra parte, la integración R-Python en SQL Server, aparte de enfatizar la productividad y la legibilidad del código, también puede aprovechar el proceso paralelo de consultas, la seguridad y un

mejor manejo de recursos.

Ahora, Microsoft ha renombrado a los *Servicios R* como *Servicios de Aprendizaje de Máquina* (Machine Learning Services), trayendo R y Python debajo del paraguas. El componente renombrado *Microsoft Machine Learning Services* permite a Python correr directamente en el servidor de la base de datos, o junto con scripts T-SQL incrustados.

Los desarrolladores han usado el procedimiento almacenado llamado `sp_execute_external_script` para correr código R, cuyo primer parámetro es `@language`. Microsoft ha diseñado este procedimiento almacenado para no tener límites fijos.

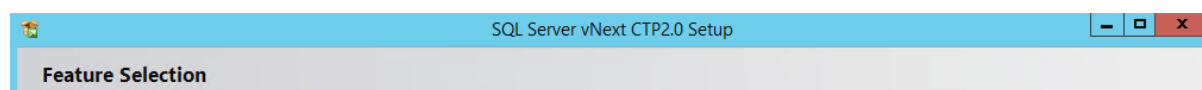
El motor hereda las características de R, para adoptar Python. Para correr código Python en SQL Server, tenemos que instalar SQL Server 2017 CTP 2.0 con la característica *Machine Learning Services with Python*. Es importante notar que otras versiones de SQL Server no soportan la integración con Python.

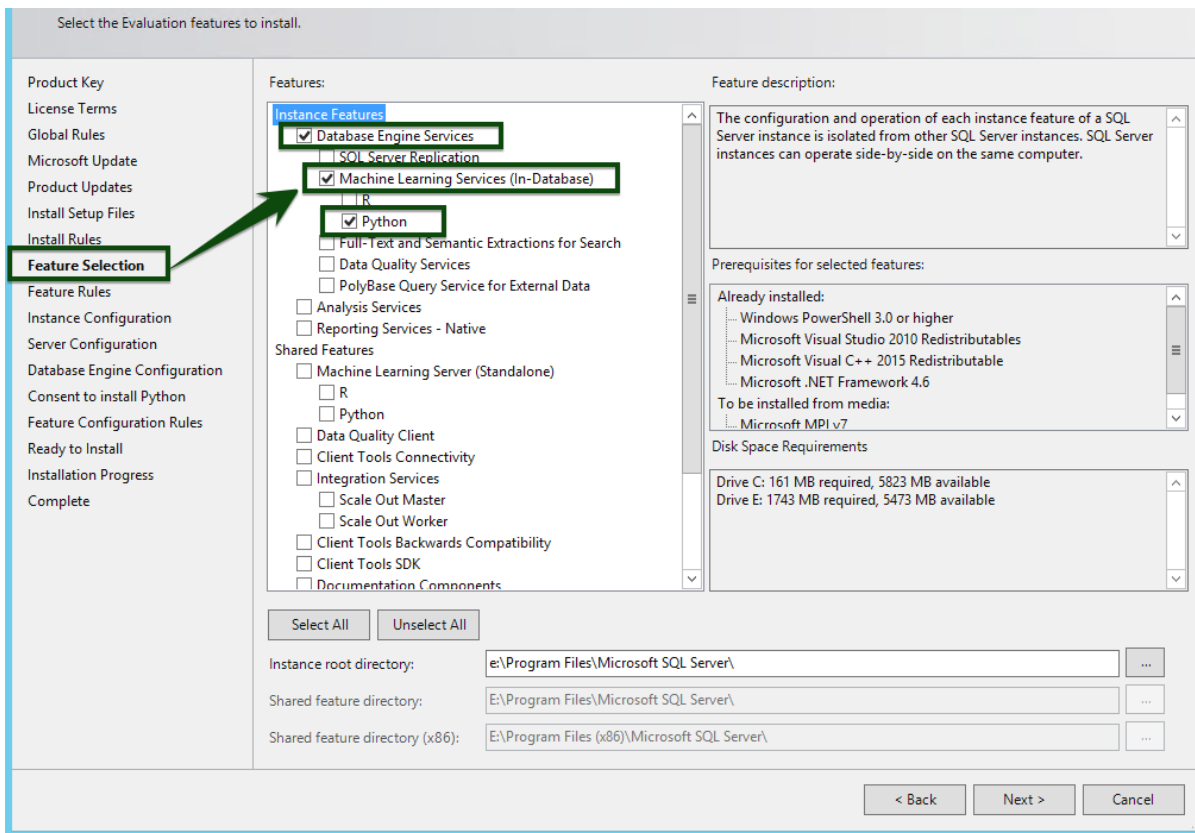
El artículo explora

- La instalación del motor de la base de datos y la configuración de *Machine Learning Services*
- La configuración de Python
- La configuración de la instancia para permitir la ejecución de scripts que usan un ejecutable externo
- Soporte para tipos de datos
- Ejemplos para entender el uso de Python en SQL

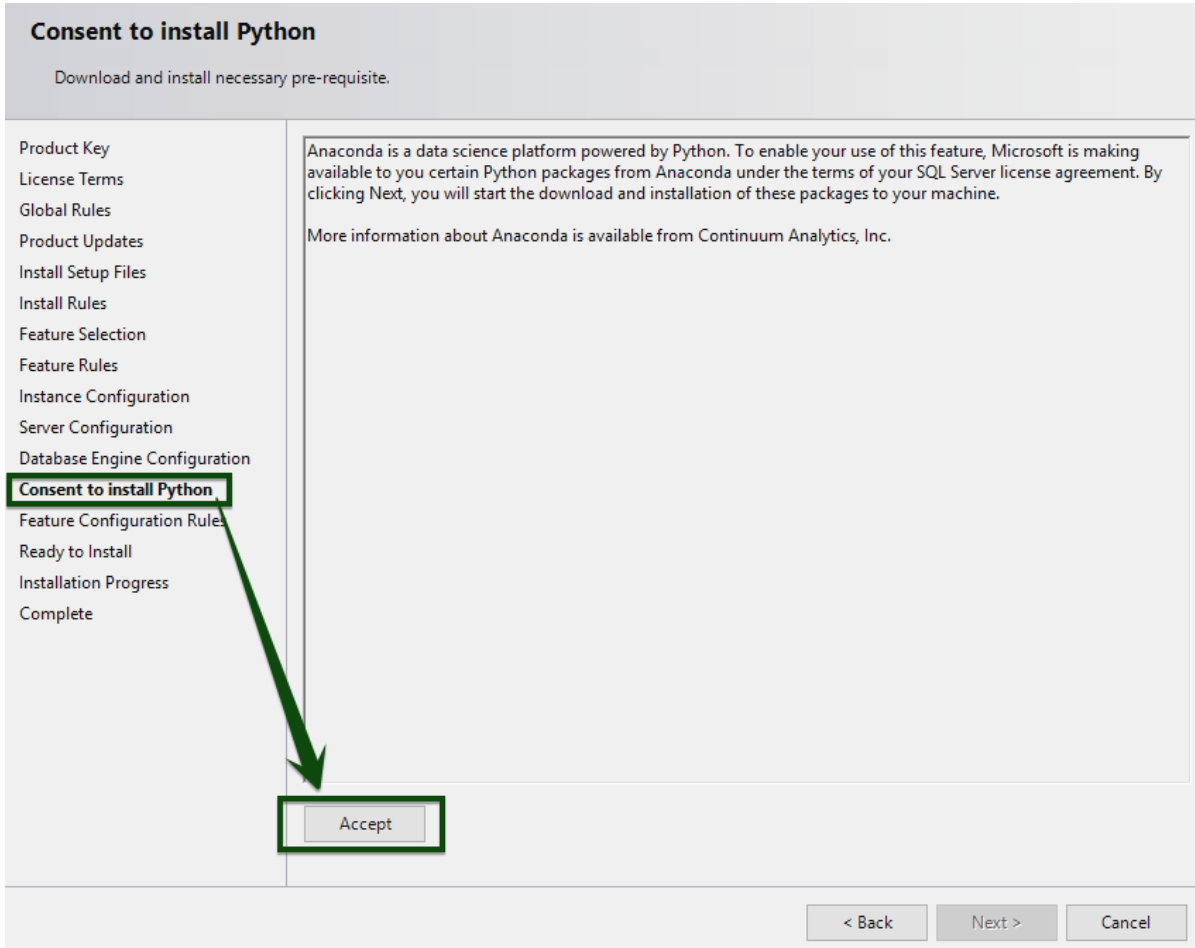
Instalación

- Descargue SQL Server 2017 CTP 2.0. Durante la Selección de Características (Feature Selection) en la instalación, Python está listado como parte de Machine Learning Services.
- Ejecute el asistente de configuración para SQL Server 2017.
- En la pestaña de instalación, haga clic en New SQL Server stand-alone installation or add features to an existing installation.
- En la página Feature Selection, seleccione las opciones que ve en la captura de pantalla.
 - **Database Engine Services:** Para usar Python con SQL Server, usted debe instalar una instancia del motor de la base de datos.
 - **Machines Services (In-Database):** Esta opción instala los servicios de la base de datos que soportan la ejecución de scripts Python.
- **Python:** Elija esta opción para obtener el ejecutable de Python 3.5 y seleccione las librerías desde la distribución Anaconda

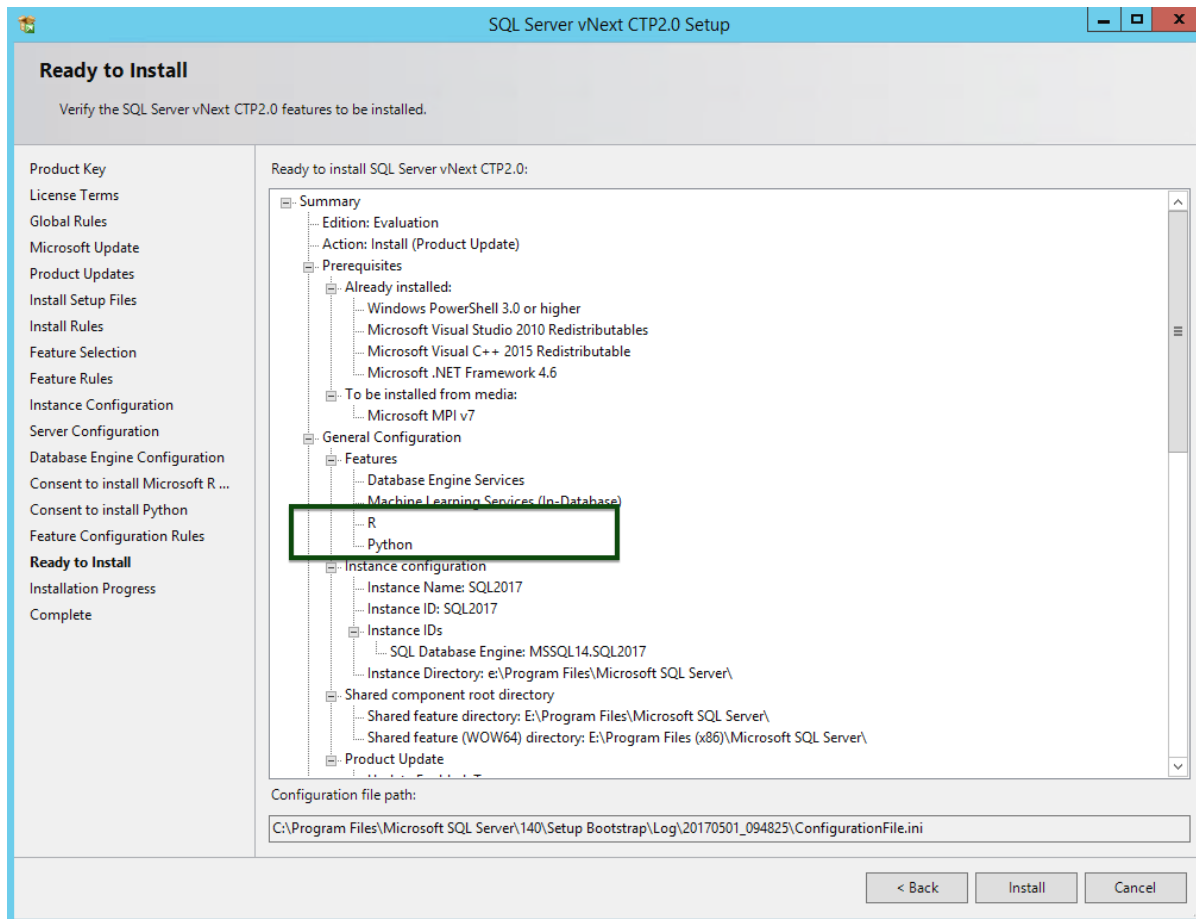




En la página de abajo, **Consent to Install Python**, haga clic en **Accept**.



En la página **Ready to Install**, verifique que los componentes seleccionados están incluidos



Después de una instalación exitosa, la instancia está lista para habilitar un parámetro de ejecución de script externo.

Para habilitar la Instancia SQL para correr scripts de Python:

- Abra SQL Server Management Studio.
- Conéctese a la instancia donde Machine Learning Services está instalado, y ejecute el siguiente comando:

sp_configure

- Para habilitar la característica de scripts externos que soportan Python, ejecute la siguiente sentencia:

```
EXEC sp_configure 'external scripts enabled', 1
RECONFIGURE WITH OVERRIDE
```

- Reinicie la Instancia SQL.

EXECUTE sp_configure					
100 %					
Results Messages					
	name	minimum	maximum	config_value	run_value
25	default language	0	9999	0	0
26	default trace enabled	0	1	1	1
27	disallow results from triggers	0	1	0	0
28	FKM provider enabled	0	1	0	0
29	external scripts enabled	0	1	1	1
30	filestream access level	0	2	0	0
31	fill factor (%)	0	100	0	0

sp_execute_external_script

sp_execute_external_script es un procedimiento almacenado que se ejecuta con un script R/Python provisto como un argumento. Para habilitar el funcionamiento notmal de este procedimiento almacenado externo, usted debe tener acceso de administrador a su instancia de SQL Server, de modo que pueda correr el comando sp_configure. El procedimiento invocará el servicio de plataforma de lanzamiento a la respectiva librería para su ejecución.

```
sp_execute_external_script
    @language = N'language' ,
    @script = N'script',
    @input_data_1 = ] 'input_data_1'
    [ , @input_data_1_name = ] N'input_data_1_name' ]
    [ , @output_data_1_name = 'output_data_1_name' ]
    [ , @parallel = 0 | 1 ]
    [ , @params = ] N'@parameter_name data_type [ OUT | OUTPUT ] [ ,...n ]'
    [ , @parameter1 = ] 'value1' [ OUT | OUTPUT ] [ ,...n ]
    [ WITH <execute_option> ]
[;]

<execute_option>::=
{
    { RESULT SETS UNDEFINED }
    | { RESULT SETS NONE }
    | { RESULT SETS ( <result_sets_definition> ) }
}
```

Parámetro	Propósito
@language = N'Python'	Parámetro de lenguaje de scripting, en este caso es Python
@script = N' `	Cuerpo del script Python
@input_data_1 = N' T-SQL Statement'	La sentencia T-SQL que lee datos desde la tabla SQL

```
@output_data_1_name = N' Data  
Frame Name'
```

Contiene el marco de datos generado dentro del Script Python

```
WITH RESULT SETS ((Col1  
DataType,Col2 DataType ))
```

Especifica la columna Output y Datatype de las columnas de marco de datos. Esto es opcional.

Ejecute Código Python en SQL Server

Microsoft ha hecho posible incrustar código Python directamente en bases de datos SQL Server incluyendo el código como un procedimiento almacenado de T-SQL.

Soporte de Tipo de Datos

Python soporta un número limitado de tipos de datos en comparación a SQL Server. Como resultado, cuando sea que usted use datos desde SQL Server en Script Python, los datos pueden ser implícitamente convertidos a un tipo compatible con Python. De todas maneras, a menudo una conversión exacta no puede ser realizada automáticamente, y un error es retornado. Esta tabla lista las conversiones implícitas que son provistas. Otros tipos de datos no son soportados.

TIPO SQL	TIPO PYTHON
bigint	numeric
binary	raw
bit	bool
char	str
float	float64
int	int32
nchar	str
nvarchar	str
nvarchar(max)	str
real	float32
smallint	int16
tinyint	uint8

varbinary	bytes
varbinary(max)	bytes
varchar(n)	str
varchar(max)	str

Ejemplos

Imprimir el valor de entrada

```
exec sp_execute_external_script
@language =N'Python',
@script=N'OutputDataSet = InputDataSet
print("Input data is {0}".format(InputDataSet))
',
@input_data_1 = N'SELECT 1 as col'
```

Encontrar la media de una lista dada

```
execute sp_execute_external_script
@language = N'Python',
@script = N'
l = [15, 18, 2, 36, 12, 78, 5, 6, 9]
print(sum(l) / float(len(l)))
'
```

```
exec sp_execute_external_script
@language =N'Python',
@script=N'OutputDataSet = InputDataSet
print("Input data is {0}".format(InputDataSet))
',
@input_data_1 = N'SELECT 1 as col'

execute sp_execute_external_script
@language = N'Python',
@script = N'
l = [15, 18, 2, 36, 12, 78, 5, 6, 9]
print(sum(l) / float(len(l)))
'
```

Results

```
(1 row(s) affected)
STDOUT message(s) from external script:
E:\Program Files\Microsoft SQL Server\MSSQL14.SQL2017\PYTHON_SERVICES\lib\site-packages\revoscalepy
Input data is      col
0      1

STDOUT message(s) from external script:
E:\Program Files\Microsoft SQL Server\MSSQL14.SQL2017\PYTHON_SERVICES\lib\site-packages\revoscalepy
20.111111111111111
```

Operador de formato

```

exec sp_execute_external_script
@language = N'Python'
,@script=
N'
print("""jan:{2} feb:{0} mar:{2} Apr:{1} May:{2} Jun:{1} Jul:{2} Aug:{2} Sep:{1}
Oct:{2} Nov:{1} Dec:{2}""").format(InputDataSet.A, InputDataSet.B, InputDataSet.
C))
',
@input_data_1 = N'select 28 as A ,30 as B,31 as C'

```

```

exec sp_execute_external_script
@language = N'Python'
,@script=
N'
print("""jan:{2} feb:{0} mar:{2} Apr:{1} May:{2} Jun:{1} Jul:{2} Aug:{2} Sep:{1} Oct:{2} Nov:{1} Dec:{2}""").format(InputDataSet.A, InputDataSet.B, InputDataSet.C))
',
@input_data_1 = N'select 28 as A ,30 as B,31 as C'

```

Messages

STDOUT message(s) from external script:
E:\Program Files\Microsoft SQL Server\MSSQL14.SQL2017\PYTHON_SERVICES\lib\site-packages\revoscalepy
jan:0 31
Name: C, dtype: int32 feb:0 28
Name: A, dtype: int32 mar:0 31
Name: C, dtype: int32 Apr:0 30
Name: B, dtype: int32 May:0 31
Name: C, dtype: int32 Jun:0 30
Name: B, dtype: int32 Jul:0 31
Name: C, dtype: int32 Aug:0 31
Name: C, dtype: int32 Sep:0 30
Name: B, dtype: int32 Oct:0 31
Name: C, dtype: int32 Nov:0 30
Name: B, dtype: int32 Dec:0 31
Name: C, dtype: int32

Usando bucles y ramas

```

execute sp_execute_external_script
@language = N'Python',
@script = N'
for i in range(5):
    if i<3 :
        print("i is now:", i*2)'

```

```

execute sp_execute_external_script
@language = N'Python',
@script = N'
for i in range(5):
    if i<3 :
        print("i is now:", i*2)'

```

Messages

STDOUT message(s) from external script:
E:\Program Files\Microsoft SQL Server\MSSQL14.SQL2017\PYTHON_SERVICES\lib\site-packages\revoscalepy
i is now: 0
i is now: 2
i is now: 4

Pasar una tabla como entrada y generar una columna computada llamada bonus

- Cree la tabla EMP
- Inserte valores de prueba
- Ejecute el script Python para generar la columna computada

```
DROP TABLE IF EXISTS dbo.EMP
GO
CREATE TABLE [dbo].[EMP] (
    [empno] [int] NOT NULL,
    [ename] [varchar](10) NULL,
    [job] [varchar](9) NULL,
    [mgr] [int] NULL,
    [hiredate] [datetime] NULL,
    [sal] float NULL,
    [comm] [numeric](7, 2) NULL,
    [dept] [int] NULL,
    PRIMARY KEY CLUSTERED
    (
        [empno] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]

GO

INSERT INTO EMP VALUES
(1, 'Prashanth', 'ADMIN', 6, '12-17-1990', 18000, NULL, 4)
INSERT INTO EMP VALUES
(2, 'Jayaram', 'MANAGER', 9, '02-02-1998', 52000, 300, 3)
INSERT INTO EMP VALUES
(3, 'thanVitha', 'SALES I', 2, '01-02-1996', 25000, 500, 3)

SELECT EMPNO, ENAME, SAL from EMP

EXECUTE sp_execute_external_script
@language = N'Python',
@script=N'OutputDataSet = InputDataSet
for i in OutputDataSet["sal"]:
    OutputDataSet["Bonus"]=OutputDataSet["sal"]*0.05',
@input_data_1 = N'SELECT [empno],[ename],sal,0 as Bonus from EMP'
WITH RESULT SETS ((EMPNO int, ENAME varchar(10), SAL float, Bonus float))
```

```
DROP TABLE IF EXISTS dbo.EMP
GO
CREATE TABLE [dbo].[EMP](
    [empno] [int] NOT NULL,
    [ename] [varchar](10) NULL,
    [job] [varchar](9) NULL,
    [mgr] [int] NULL,
    [hiredate] [datetime] NULL,
    [sal] float NULL,
    [comm] [numeric](7, 2) NULL,
    [dept] [int] NULL,
    PRIMARY KEY CLUSTERED
    (
        [empno] ASC
    )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO

INSERT INTO EMP VALUES
(1, 'Prashanth', 'ADMIN', 6, '12-17-1990', 18000, NULL, 4)
INSERT INTO EMP VALUES
(2, 'Jayaram', 'MANAGER', 9, '02-02-1998', 52000, 300, 3)
INSERT INTO EMP VALUES
(3, 'thanVitha', 'SALES I', 2, '01-02-1996', 25000, 500, 3)

SELECT EMPNO, ENAME, SAL from EMP
```



```

EXECUTE sp_execute_external_script
@language = N'Python',
@script=N'OutputDataSet = InputDataSet
for i in OutputDataSet["sal"]:
    OutputDataSet["Bonus"]=OutputDataSet["sal"]*0.05',
@input_data_1 = N'SELECT [empno], [ename], sal @ as Bonus from EMP'
WITH RESULT SETS ((EMPNO int, ENAME varchar(10), SAL float, Bonus float))

```

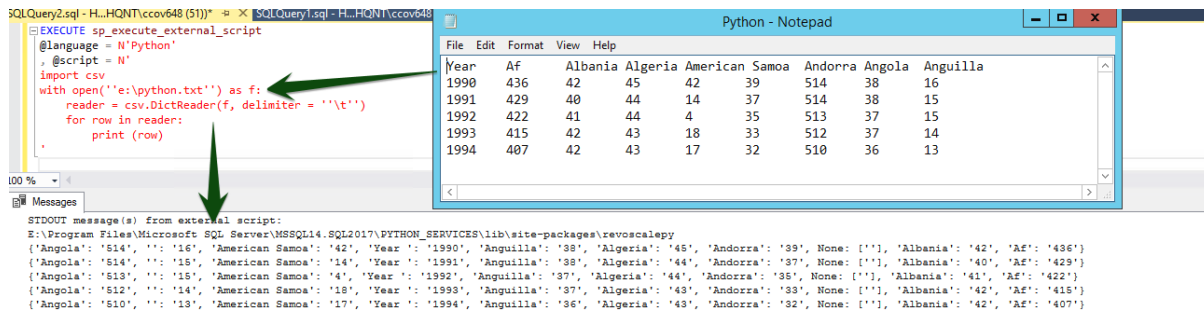
EMPNO	ENAME	SAL	Bonus
1	Prashanth	18000	900
2	Jayaram	52000	2600
3	thanVitha	25000	1250

Leer contenido desde un archivo de texto usando la función DictReader.

```

EXECUTE sp_execute_external_script
@language = N'Python'
, @script = N'
import csv
with open('e:\python.txt') as f:
    reader = csv.DictReader(f, delimiter = '\t')
    for row in reader:
        print (row)
'

```



```

EXECUTE sp_execute_external_script
@language = N'Python'
, @script = N'
import csv
with open('e:\python.txt') as f:
    reader = csv.DictReader(f, delimiter = '\t')
    for row in reader:
        print (row)
'

```

Year	Af	Albania	Algeria	American Samoa	Andorra	Angola	Anguilla
1990	436	42	45	42	39	514	38
1991	429	40	44	14	37	514	38
1992	422	41	44	4	35	513	37
1993	415	42	43	18	33	512	37
1994	407	42	43	17	32	510	36

El siguiente ejemplo muestra cómo importar la librería Pandas para obtener acceso a los marcos de datos y hacer las siguientes tareas

- Lea el archivo CSV usando la librería Pandas
- Encuentre el número de filas usando el objeto de Pandas pyo
- Recupere la primera fila del archivo csv
- Calcule la media de cada columna estadística

```

EXECUTE sp_execute_external_script
@language = N'Python'
, @script = N'
import pandas
pyo = pandas.read_csv("e:\InputServer_1.csv")
print (pyo)
#Finding the number of rows
print (pyo.shape)

```

```
#Looking at the first row of the data
print (pyo.head(1))
#Find the average of each statistic
print (pyo.mean())
'
```

The screenshot shows a SQL Server Enterprise Manager window with a Python script executed. The script reads a CSV file from 'e:\InputServer_1.csv'. The output in the Messages pane shows the CSV data and its statistics. Annotations with arrows point to specific parts of the output:

- An arrow points from the script line `pyo = pandas.read_csv("e:\InputServer_1.csv")` to the CSV data output.
- An arrow points from the script line `print (pyo.head(1))` to the first row of the CSV data output.
- An arrow points from the script line `print (pyo.mean())` to the mean values of the numeric columns.

The CSV data output is as follows:

ServerName	Drive	LowTh	WarnTh	CritTh
HQDBSP19	E:	8	5	3
HQDBSP18	F:	8	5	3
HQDBSP19	G:	20	15	3
APMESDP02	E:	10	8	5
APMESDP02	F:	20	15	10

The mean values output is as follows:

```

(5, 5)
ServerName Drive LowTh WarnTh CritTh
0 HQDBSP19 E: 8 5 3
LowTh 13.2
WarnTh 9.6
CritTh 4.8
dtype: float64

```

Annotations:

- To display no of Rows
- To display first row of the csv file
- Mean of the numeric data

Encuentre la media sobre la columna estadística

Los datos son alimentados desde la tabla y la computación ocurre usando la librería Pandas

```
DROP TABLE IF EXISTS MyData;
CREATE TABLE MyData([Col1] INT NOT NULL) ON [PRIMARY];
INSERT INTO MyData VALUES(1);
INSERT INTO MyData VALUES(10);
INSERT INTO MyData VALUES(100);
GO

-- print all rows of MyData table
SELECT * FROM MyData;

--Find mean of Col1

EXECUTE sp_execute_external_script
@language = N'Python'
, @script = N'
import pandas
print("*****")
OutputDataSet = InputDataSet
print (OutputDataSet)
print (OutputDataSet.mean())
print("*****")
'
, @input_data_1 = N'SELECT * from Mydata'
```

The screenshot shows a SQL Server Enterprise Manager window with a SQL script executed. The script creates a table, inserts data, and executes a Python script to calculate the mean of a column. The output in the Messages pane shows the results of the SQL and Python execution.

The SQL script is as follows:

```

DROP TABLE IF EXISTS MyData;
CREATE TABLE MyData([Col1] INT NOT NULL) ON [PRIMARY];
INSERT INTO MyData VALUES(1);
INSERT INTO MyData VALUES(10);
INSERT INTO MyData VALUES(100);
GO

```

The Python script is as follows:

```

EXECUTE sp_execute_external_script
@language = N'Python'
, @script = N'
import pandas
print("*****")
OutputDataSet = InputDataSet
print (OutputDataSet)
print (OutputDataSet.mean())
print("*****")
'
, @input_data_1 = N'SELECT * from Mydata'

```

The output in the Messages pane is as follows:

```

(5, 5)
ServerName Drive LowTh WarnTh CritTh
0 HQDBSP19 E: 8 5 3
LowTh 13.2
WarnTh 9.6
CritTh 4.8
dtype: float64

```

```
-- print all rows of MyData table
SELECT * FROM MyData;

EXECUTE sp_execute_external_script
@language = N'Python'
, @script = N'
import pandas
print("*****")
OutputDataSet = InputDataSet
print (OutputDataSet)
print (OutputDataSet.mean())
print("*****")
'
, @input_data_1 = N'SELECT * from Mydata'
```

100 %

Results Message

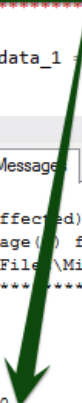
(3 row(s) affected)

STDOUT message(s) from external script:

E:\Program Files\Microsoft SQL Server\MSSQL14.SQL2017\PYTHON_SERVICES\lib\site-packages\revoscalepy

Col1
0 1
1 10
2 100

Col1 37.0
dtype: float64



Resumen

Este artículo cubrió cómo SQL Server 2017 introduce soporte para análisis de datos, y el uso de Python en adición a los scripts R. He detallado los procedimientos de instalación, los detalles de configuración y la ejecución de un ejemplo de script Python SQL.

La estrecha integración de R/Python a la máquina de SQL Server elimina el movimiento innecesario de datos a través de las máquinas; piense en mover millones/billones de filas al cliente para modelar y tantear sobre la red – es un trabajo complicado y tedioso. Esta es una de las razones por qué los científicos de datos se apoyan en muestras (Test set v/s Train set). Es un enfoque útil, especialmente donde hay problemas de soberanía de datos y falta de cumplimiento de requerimientos. Su código corre dentro de los límites de seguridad de SQL Server, gatillado por una sola llamada desde los procedimientos almacenados T-SQL.

El siguiente artículo en esta serie:

- [Interpolación y transformación de Datos usando Python en SQL Server 2017](#)

Vea más

Considere estas [herramientas gratis para SQL Server](#) que mejoran la productividad del desarrollador de bases de datos.

Referencias

- Aprendizaje de máquina mejorado para SQL Server 2017 (TechNet)
- Soporte para Tipos de Datos (Microsoft Docs)
- Para descargar SQL Server 2017 usted puede Registrarse y Descargar la versión Completa o la versión de evaluación gratis (180 días)—(Microsoft))
- Correr Python usando T-SQL (Microsoft Docs)



Prashanth Jayaram

Soy un experto en tecnologías con más de 11 años de experiencia en tecnologías de base de datos. Soy Microsoft Certified Professional y tengo el respaldo de una Licenciatura en Master en aplicaciones de computadoras.

Mi especialidad es el diseño y la implementación de soluciones de alta disponibilidad y la migración de bases de datos multiplataforma. Las tecnologías en las que trabajo actualmente son SQL Server, PowerShell, Oracle y MongoDB.

[Ver todas las publicaciones de Prashanth Jayaram](#)

Related Posts:

1. [Cómo sincronizar bases de datos SQL de Azure y bases de datos locales con sincronización de datos SQL](#)
2. [Cómo importar/exportar datos a SQL Server utilizando el Asistente para importación y exportación de SQL Server](#)
3. [Cómo poder importar los datos de un archivo de Excel a una base de datos de SQL Server](#)
4. [Una guía para Administradores de Bases de Datos para resolución de problemas de SQL Server – Parte 1 – Métricas de problemas y desempeño](#)
5. [Usando paquetes SSIS para importar datos de MS Excel en una base de datos](#)

Diseño de bases de datos SQL

4,257 Views



Start the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS 

Name

Be the first to comment.

ALSO ON SQL SHACK

SQL Server replication: Configuring Snapshot and Transactional Replication

1 comment • 5 months ago



André Coelho — I am having trouble with the Subscriber configuration on STEP 4. I Cannot find ou connect my remote SQL Instance (It is

SQL Server monitoring tools for memory performance

1 comment • 5 months ago



manu — Nicely explained. DMV named sys.dm_os_process_memory (introduced in SQL 2008) definitely helps by providing

SQL ISNULL function

1 comment • 3 months ago



Lokesh Vij — Example 1: SQL Server ISNULL function in an argumentOutput for this example should be 'SQLServer' instead of NULL

SQL Server in Azure Kubernetes Service (AKS)

2 comments • 3 months ago



Sudheer — Great article. Steps are in detail