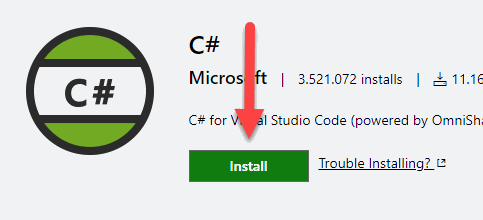
**▪️4.5.- Creación de clases**

**Instalando las extensiones necesarias**

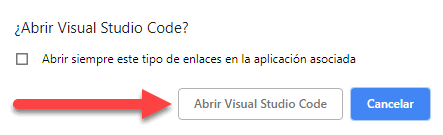
Para poder desarrollar todo su potencial, Visual Studio Code utiliza un sistema de extensiones que nos permiten ampliar su funcionalidad. Estas extensiones se pueden descargar desde el propio entorno o desde el [Extension Marketplace](https://code.visualstudio.com/docs/editor/extension-gallery" \t "_blank) (aunque esto realmente, lo que va a hacer es abrir el IDE y llevarnos a la extensión).

Para empezar a preparar nuestro entorno, vamos a utilizar la extensión para el lenguaje [C#](https://marketplace.visualstudio.com/items?itemName=ms-vscode.csharp).

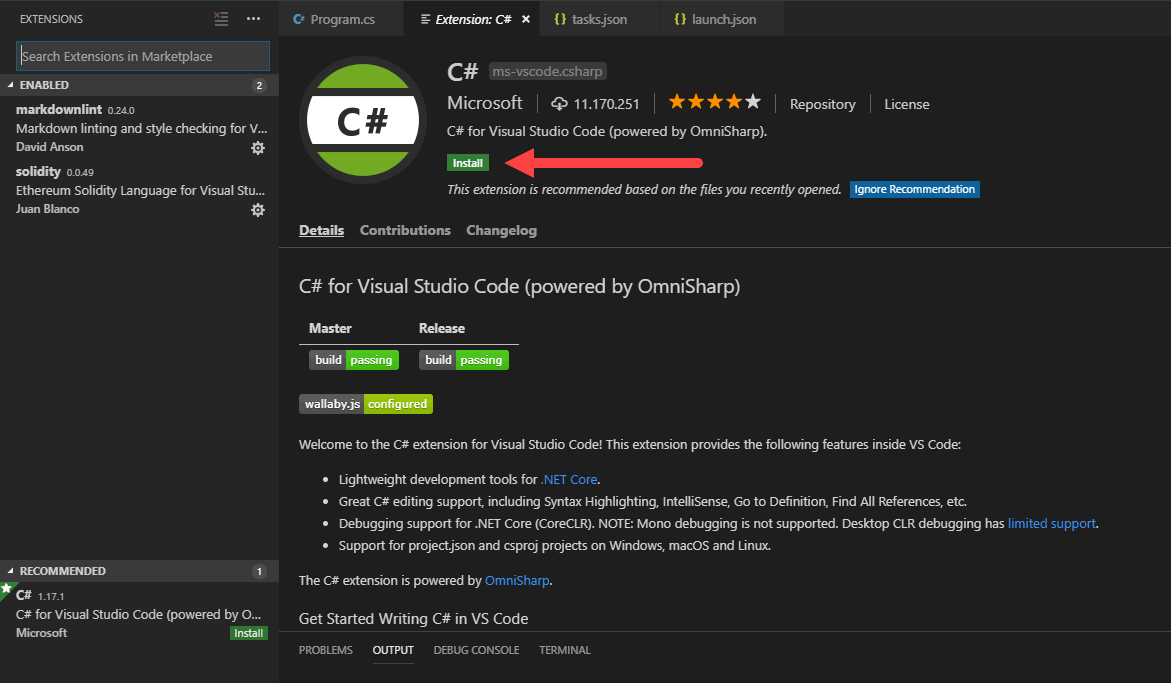
Para instalar esta extensión, desde la web, basta con que pulsemos sobre Install:



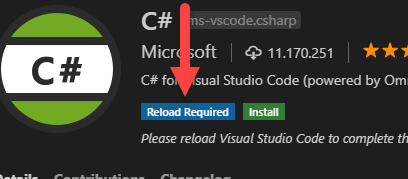
Nos va a pedir como requisito tener instalado Visual Studio Code, y como nosotros ya lo tenemos, en la ventana emergente pulsamos sobre el botón Abrir Visual Studio Code:



Esto nos abre Visual Studio Code, y nos muestra directamente la extensión, aquí basta con pulsar sobre el botón Install para que se inicie el proceso:



Una vez que termine tenemos que recargar el IDE. Para eso basta con pulsar en el botón de recargar:



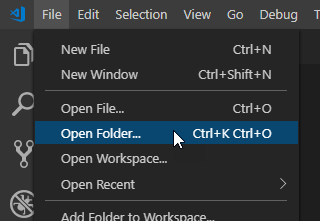
Con esto, ya tenemos nuestro entorno de trabajo Visual Studio Code para trabajar con el lenguaje C# y con .NET Core.

Podríamos haberlo buscado directamente desde el entorno e instalarlo desde allí más fácilmente, ahorrándonos un par de pasos.

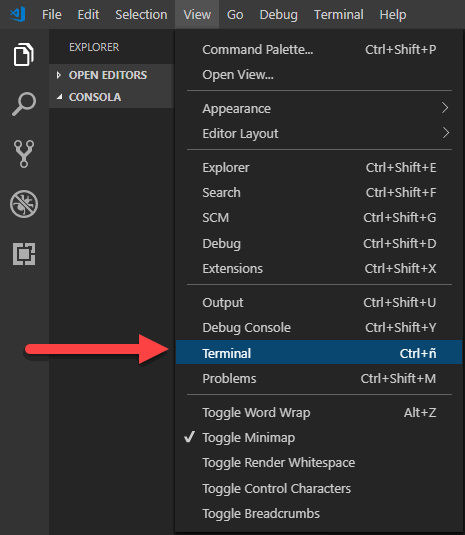
Vamos a ver cómo poder ejecutar un proyecto con nuestro nuevo entorno.

**Creando una aplicación de consola .NET Core en Visual Studio Code**

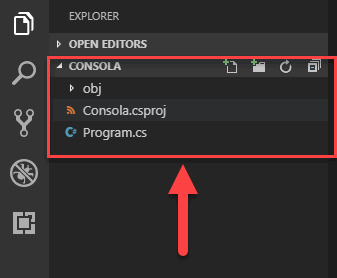
Lo primero que vamos a necesitar es crear una carpeta en la que queramos que esté el proyecto. Una vez que la tengamos vamos a abrir en Visual Studio Code la carpeta que acabamos de crear, utilizando el menú File·Open Folder o con el acceso directo Ctrl+K,Ctrl+O (son dos combinaciones de teclas, una después de la otra):



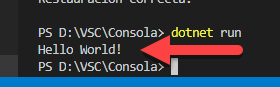
Una vez que tengamos la carpeta, vamos a necesitar sacar una ventana de terminal para lanzar comandos de compilación y similares. Para eso, dentro del menú View, pulsamos sobre el botón Terminal o con el acceso directo Ctrl+ñ:



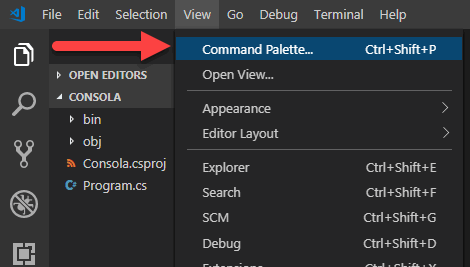
Ahora dentro de esa terminal vamos a ejecutar el comando dotnet new console para crear nuestro proyecto (éste tomará el nombre de la carpeta en la que estamos). Cuando acabe la ejecución, podemos ver que han aparecido los ficheros del proyecto en el explorador:



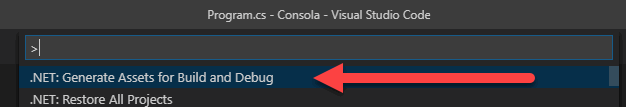
Con esto, si ejecutamos desde el terminal el comando dotnet run, podemos ver como la aplicación se ejecuta sin ningún problema:



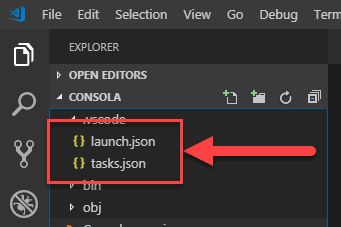
Con lo que hemos hecho podríamos modificar el código y ejecutarlo con los cambios, pero aún nos falta una parte importante: **poder depurarlo**, como lo haríamos en Visual Studio. Para eso tenemos que añadir algunos ficheros JSON mediante el asistente para que así se configuren correctamente también. Vete al menú View·Command Palette o usa el acceso directo Ctrl+Shift+P:



Y dentro del desplegable que aparece debemos buscar la opción .NET: Generate Assets for Build and Debug:



Esto nos generará los ficheros JSON necesarios para poder **compilar, ejecutar y depurar** el proyecto desde Visual Studio Code.

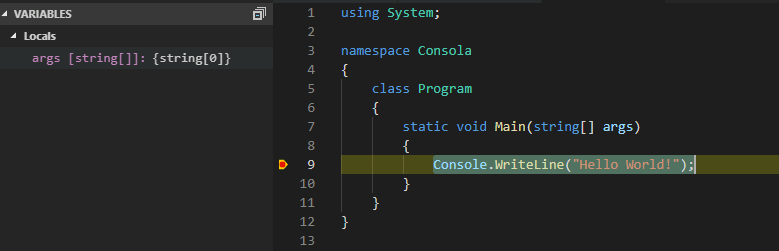


Una vez que tenemos todo esto, ya solo nos queda probar que funciona, para ello, vamos a poner un punto de interrupción situándonos sobre la línea:

Console.WriteLine("Hello World!");

y pulsando la tecla F9.

Si ahora ejecutamos el proyecto (pulsando F5) podemos ver que, efectivamente, el punto de interrupción se aplica y la ejecución se detiene ahí, permitiéndonos depurar el proyecto:

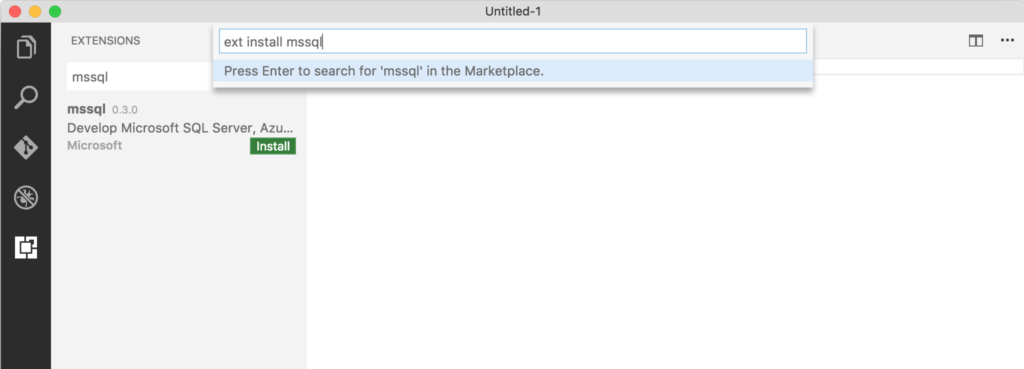


4.6.- Acceso a datos

Estos días he estado dando una sesión de desarrollo en uno de mis clientes y, por motivos de viajes y eventos, he tenido que llevarme mi Mac en lugar de mi Surface. Aún así, quería demostrar a los asistentes cómo de sencillo era trabajar con una base de datos en Azure SQL Database desde su propia máquina, pero no tenía disponible SQL Server Management Studio al estar usando Mac. Por ello estuve viendo posibles herramientas y formas de conectarme, sin que ello me generara un gasto, y aquí es cuando me topé con la [**extensión mssql para Visual Studio Code**](https://marketplace.visualstudio.com/items?itemName=ms-mssql.mssql), la cual me permite conectarme a mi base de datos desde el propio editor.

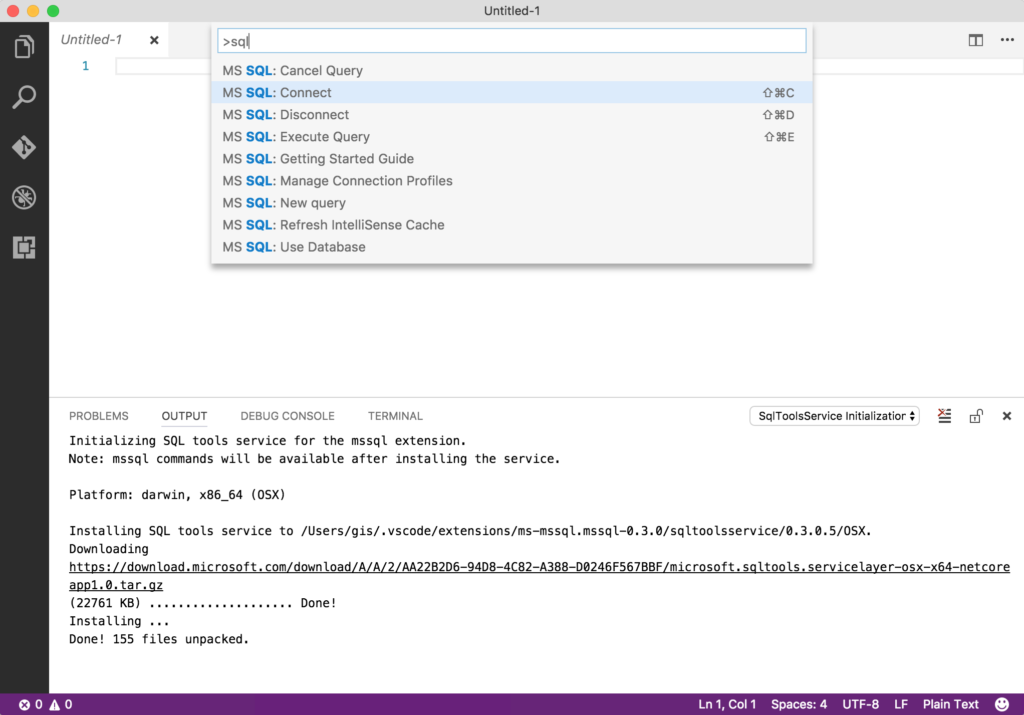
Lo primero que debes hacer es instalar dicha extensión usando ⌘+P y escribiendo el siguiente comando:

|  |  |
| --- | --- |
| 1 | ext install mssql |

[[](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-ext-install-mssql.png)](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-ext-install-mssql.png)

Visual Studio Code – ext install mssql

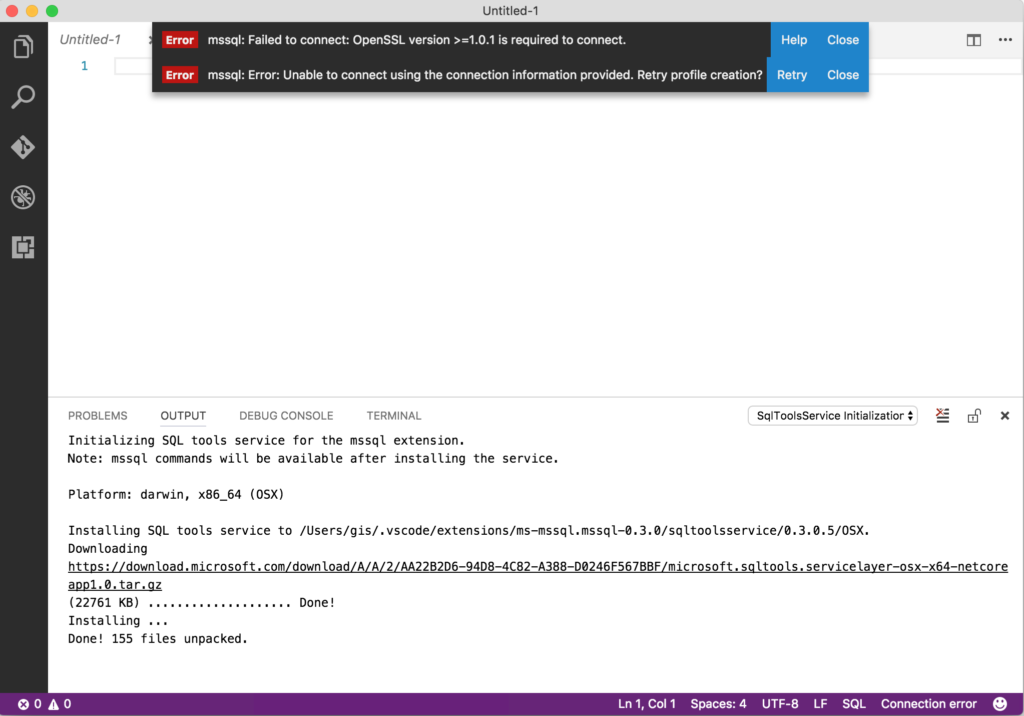
Ahora ya tienes los comandos necesarios para hacer una conexión con tu base de datos. El siguiente paso es usar **⌘+P >sqlconnect** para crear un perfil de conexión. Al hacer esto se cargará un nuevo paquete y te pedirá que elijas el tipo de lenguaje que vas a usar sobre el documento en blanco que tienes abierto, que debe ser **SQL**.

[[](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-sqlconnect-command.png)](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-sqlconnect-command.png)

Visual Studio Code – sqlconnect command

Durante la creación del perfil se te pedirán los datos típicos: dirección, nombre o IP del servidor donde está la base de datos, nombre de la base de datos, usuario y contraseña.

Sin embargo, es posible que cuando intentes conectarte con el perfil que acabas de crear te encuentres con el siguiente error:

[[](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-mssql-failed-to-connect-OpenSSL.png)](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-mssql-failed-to-connect-OpenSSL.png)

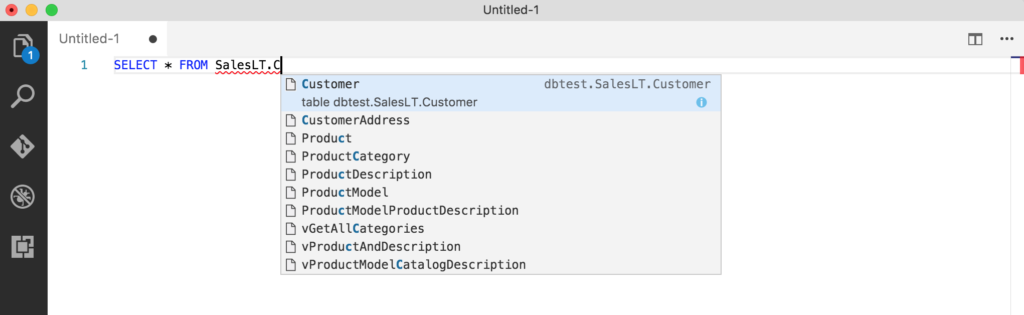
Visual Studio Code – mssql failed to connect OpenSSL

Para solucionarlo basta con seguir los pasos que aparecen en este enlace: <https://github.com/Microsoft/vscode-mssql/wiki/OpenSSL-Configuration>. Una vez hecho esto, cierra por completo Visual Studio Code e intenta conectarte de nuevo. Si todo ha salido bien (si estás probando con una base de datos en Azure SQL Database ¡recuerda habilitar tu IP en el Firewall!) en la parte inferior de la ventana aparecerá un mensaje que te informa de donde estás conectado.

[[](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-Azure-SQL-Database-connected.png)](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-Azure-SQL-Database-connected.png)

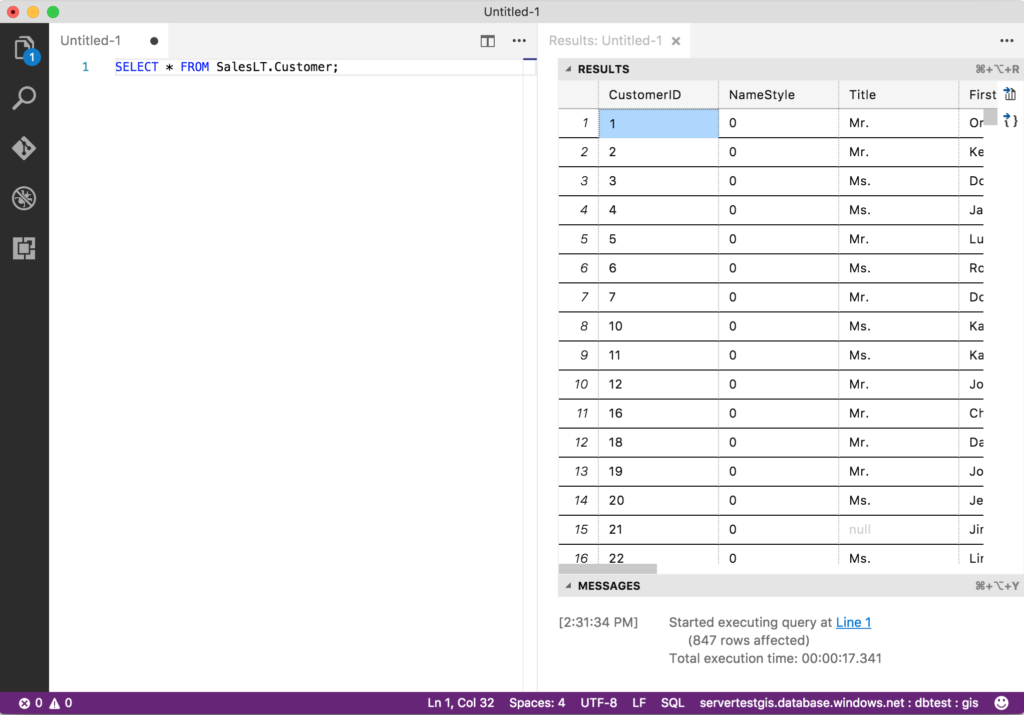
Visual Studio Code – Azure SQL Database connected

A partir de aquí, lanzar queries contra tu base de datos es tan sencillo como escribirlas en el documento en blanco donde dispones de IntelliSense con el esquema de tu base de datos.

[[](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-mssql-intellisense.png)](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-mssql-intellisense.png)

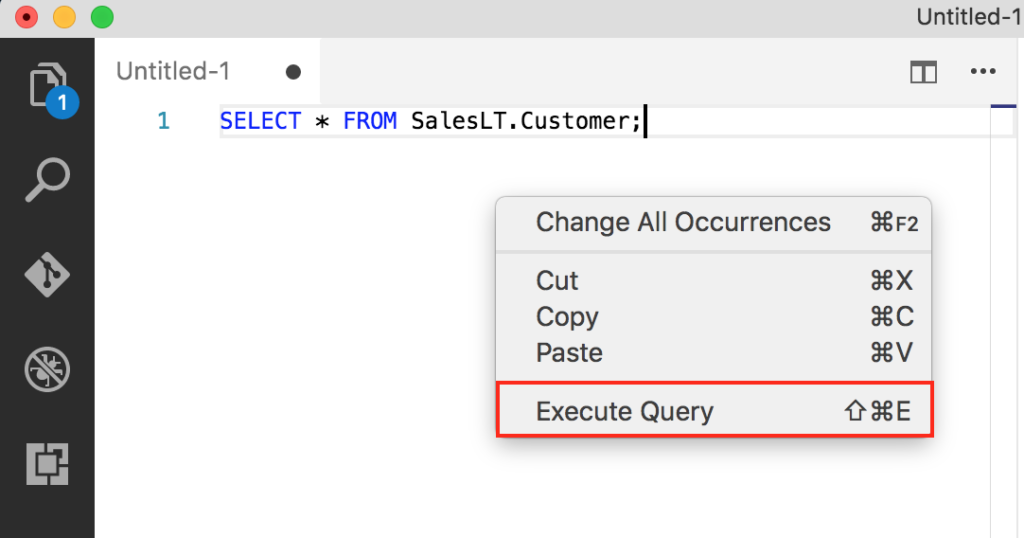
Visual Studio Code – mssql intellisense

Para ejecutar las queries pulsa **⌘+Shift+E** y el resultado se mostrará en el lado derecho de Visual Studio Code.

[[](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-mssql-results.png)](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-mssql-results.png)

Visual Studio Code – mssql – results

También es posible lanzar las queries utilizando **Control + clic** y seleccionando la opción **Execute Query**del menú contextual.

[[](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-mssql-Execute-Query.png)](https://www.returngis.net/wp-content/uploads/2017/03/Visual-Studio-Code-mssql-Execute-Query.png)

Visual Studio Code – mssql – Execute Query