Azure SQL Database 3 - VSCode를 사용하여 연결 및 쿼리

주신영 bit1010@live.com

이번 실습은 MS의 실습 자료를 참고하였습니다.

빠른 시작: Visual Studio Code를 사용하여 연결 및 쿼리를 참조하세요.

지역은 계정에 제한이 없다면 Korea Central을 선택합니다.

Visual Studio Code 설치

최신 Visual Studio Code와 mssql 확장을 설치합니다. mssql 확장 프로그램 설치에 대한 지침은 Visual Studio Code 설치 및 Visual Studio Code용 mssql을 참조하세요.

Visual Studio Code 구성

macOS의 경우 mssql 확장에서 사용하는 .NET Core의 필수 구성 요소인 OpenSSL을 설치해야 합니다. 터미널을 열고 다음 명령을 입력하여 **brew** 및 **OpenSSL**을 설치합니다.

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
brew update
brew install openssl
mkdir -p /usr/local/lib
ln -s /usr/local/opt/openssl/lib/libcrypto.1.0.0.dylib /usr/local/lib/
ln -s /usr/local/opt/openssl/lib/libssl.1.0.0.dylib /usr/local/lib/
```

서버 연결 정보 가져오기

1. Azure Portal에 로그인합니다.
2. **SQL Databases** 또는 **SQL Managed Instances** 페이지로 이동합니다.
3. **개요** 페이지에서 SQL Database의 데이터베이스에 대한 **서버 이름** 옆에 도메인으로 구성된 주소를 마우스로 해당 이름 위를 가리키고 **복사** 아이콘을 선택합니다.

언어 모드를 SQL로 설정

Visual Studio Code에서 언어 모드를 **SQL**로 설정하여 mssql 명령 및 T-SQL IntelliSense를 사용합니다.

1. 새 Visual Studio Code 창을 엽니다.
2. **Ctrl+N**을 누릅니다. 새 일반 텍스트 파일이 열립니다.
3. 상태 표시줄의 오른쪽 아래 모서리에 있는 **일반 텍스트(Plain Text)**를 선택합니다.
   * **VSCode 창의 하단 오른쪽**
4. 열리는 **언어 모드 선택** 드롭다운 메뉴에서 **SQL**을 선택합니다.

데이터베이스 연결

중요

계속하기 전에 서버 및 로그인 정보를 준비했는지 확인합니다. 연결 프로필 정보를 입력하기 시작하면 Visual Studio Code에서 포커스를 변경하는 경우 프로필 만들기를 다시 시작해야 합니다.

1. Visual Studio Code에서 **Ctrl+Shift+P**(또는 **F1** 키)를 눌러서 명령 팔레트를 엽니다.
2. **MS SQL:Connect**를 선택하고 **Enter**를 선택합니다.
3. **연결 프로필 만들기(Create Connection Profile)**를 선택합니다.
4. 프롬프트에 따라 새 프로필의 연결 속성을 지정합니다. 각 값을 지정한 후에 **Enter**를 선택하여 계속합니다.

속성	제안 값	Description
서버 이름	정규화된 서버 이름	예: **mynewserver20170313.database.windows.net**.
데이터베이스 이름	mySampleDatabase	연결할 데이터베이스입니다.
인증	SQL 로그인	이 자습서에서는 SQL 인증을 사용합니다.
사용자 이름	사용자 이름	서버를 만드는 데 사용되는 서버 관리자 계정의 사용자 이름입니다.
암호(SQL 로그인)	암호	서버를 만드는 데 사용되는 서버 관리자 계정의 암호입니다.
암호를 저장하시	Yes 또는 No	매번 암호를 입력하지 않으려면 **예**를 선택합니다.

겠습니까?		
이 프로필의 이름을 입력합니다.	**mySampleProfile**과 같은 프로필 이름	프로필을 저장할 경우 후속 로그인의 연결 속도가 빨라집니다.

성공하면 프로필이 생성되고 연결되었다는 알림이 나타납니다.

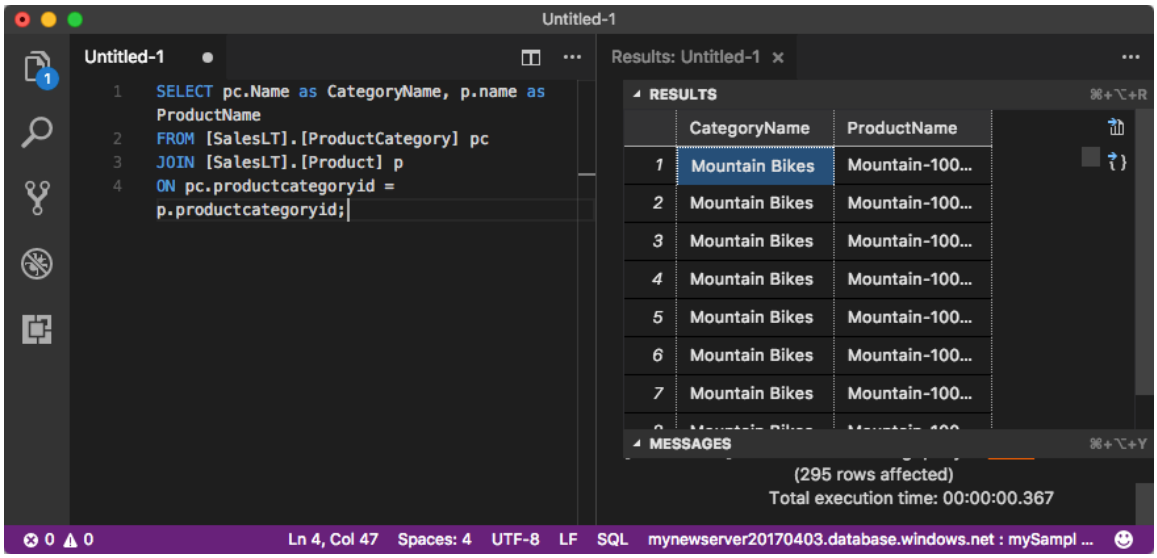
쿼리 데이터

다음 SELECT Transact-SQL 문을 실행하여 범주별 상위 20개 제품을 쿼리합니다.

1. 편집기 창에서 다음 SQL 쿼리를 붙여넣습니다.

```
SELECT pc.Name as CategoryName, p.name as ProductName
FROM [SalesLT].[ProductCategory] pc
JOIN [SalesLT].[Product] p
ON pc.productcategoryid = p.productcategoryid;
```

2. **Ctrl+Shift+E**를 눌러 쿼리를 실행하고 `Product` 및 `ProductCategory` 테이블의 결과를 표시합니다.



데이터 삽입

다음 INSERT Transact-SQL 문을 실행하여 `SalesLT.Product` 테이블에 새 제품을 추가합니다.

1. 이전 쿼리를 다음 쿼리로 바꿉니다.

```
INSERT INTO [SalesLT].[Product]
     ( [Name]
```

```
    , [ProductNumber]
    , [Color]
    , [ProductCategoryID]
    , [StandardCost]
    , [ListPrice]
    , [SellStartDate]
    )
VALUES
    ('myNewProduct'
    ,123456789
    ,'NewColor'
    ,1
     ,100
     ,100
     ,GETDATE() );
```

2. **Ctrl+Shift+E**를 눌러 `Product` 테이블에 새 행을 삽입합니다.

데이터 업데이트

다음 UPDATE Transact-SQL 문을 실행하여 추가된 제품을 업데이트합니다.

1. 이전 쿼리를 다음 쿼리로 바꿉니다.

```
UPDATE [SalesLT].[Product]
SET [ListPrice] = 125
WHERE Name = 'myNewProduct';
```

2. **Ctrl+Shift+E**를 눌러 `Product` 테이블에서 지정된 행을 업데이트합니다.
3. 위 **결과보기**의 쿼리로 확인합니다.

데이터 삭제

다음 DELETE Transact-SQL 문을 실행하여 새 제품을 제거합니다.

1. 이전 쿼리를 다음 쿼리로 바꿉니다.

```
DELETE FROM [SalesLT].[Product]
WHERE Name = 'myNewProduct';
```

2. **Ctrl+Shift+E**를 눌러 `Product` 테이블에서 지정된 행을 삭제합니다.
3. 위 **결과보기**의 쿼리로 확인합니다.