SQL With Visual Studio Code

by Simon Foster MVB · Nov. 08, 17 · Database Zone · Tutorial

Writing SQL queries is typically done with SQL Management Studio (SSMS). However, this tool is a bit of a beast, so let's look at how you could use Visual Studio Code instead.

Visual Studio Code is a free text editor — but it is so much more than *just* a text editor. Let's see how. (By the way, VS Code can be downloaded from here.)

To work with SQL Server, download the MS SQL extension. Press **CTRL** + **SHIFT** + **P**, select **Install Extension**, and type **mssql**.

Intellisense in Visual Studio Code is brilliant — better than SSMS. Let's look at how to get it all set up.

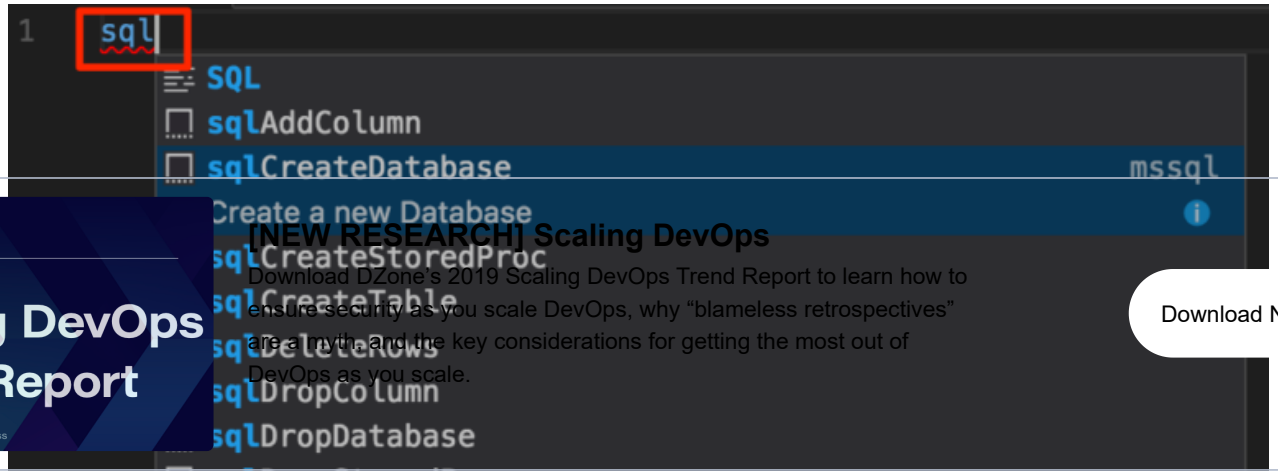
Create a new file and set the language type to SQL (press **CTRL** + **K** + **M**).

Open the command palette with **CTRL** + **SHIFT** + **P** and type SQL to show the MS SQL commands. Select the **Connect** command.

Then, select **Create Connection Profile**. This creates a profile to connect to your SQL Server. Follow the prompts to get it all set up.

Look in the bottom right corner of the status bar and you should see that you are connected!

Now, if you type **sql**, you will see a long list of SQL code snippets that you can use:



Choose a snippet to create and edit it as required. When you are happy, press **CTRL** + **SHIFT** + **E** to execute.

This is basically all there is to it! However, this is an incredibly powerful way of working; the Intellisense instantly tells you what database objects you can use in your query, and there is a wealth of different snippets you can use.

When returning data, you get a similar view to SSMS, but you can save as Excel, CSV, or JSON.

SSMS is a very graphical way of doing things. You can double-click a table and see its columns or indexes. VS Code relies on T-SQL commands, but you have access to exactly the same information.

For more information about VS Code and the MS SQL extension, check out this documentation.

Like This Article? Read More From DZone

How to Use the SQL Helper Class to Create Web APIs

Database Development in Visual Studio Using SQL Change Automation: Getting Started

How to Hide an Instance of SQL Server

Free DZone Refcard
Working With Time Series Data

Topics: DATABASE , SQL , VISUAL STUDIO , MS SQL , SSMS

On P…ow Functions

by Keshav M… …abase Zone · Tutorial

Use a golf analogy when explaining to executives.
Use a car analogy for all others. — Confucius.

See window functions more clearly...get it?

The purpose o the business and developer/business-analyst efficien

I've seen real- from hours to minutes and minutes to seconds after using window functions. Query size decreases from 40-pages to a few pages. Back in understood the business use case and created a new layer of functionality to do business reporting that included ranking, running totals, and calculating commissions and inventory based on subgroups, positions, etc. These have been in SQL standard in 2003. Every BI layer (like Tableau,

Looker, Cognos) exploits this functionality.

You may also like: Fun With SQL: Window Functions in Postgres

Introduction to Window Functions

Imagine you have scores of six golfers through two rounds. Now, you need to create the leaderboard and rank them. Rank them using SQL.

Player	Round1	Round2
Marco	75	73
Johan	72	68
Chang	67	76
Isha	74	71
Sitaram	68	72
Bingjie	71	67

Insert the data into Couchbase.

```
INSERT INTO golf
VALUES("KP1", {"player": "Marco", "round1":75, "round2":73}),
VALUES("KP2", {"player": "Johan", "round1":72, "round2":68}),
VALUES("KP3", {"player": "Chang", "round1":67, "round2":76}),
VALUES("KP4", {"player": "Isha", "round1":74, "round2":71}),
VALUES("KP5", {"player": "Sitaram", "round1":68, "round2":72}),
VALUES("KP6", {"player": "Bingjie", "round1":71, "round2":67});
```

WITHOUT window functions (current state — Couchbase 6.0)

To write the qu nctions, you need a subquery to calculate the rank for each player. This subquery has to scan through all of the data, resulting in the **O(N^2)**, which dramatically increases the execution time and throughput.


```
WITH g1 a                                    from golf)
SELECT                                            AS player,
                                                  AS r,
                                                  AS ToPar,

          (select raw 1+COUNT(*)
```

```
            from g1 as g2
              where (g2.round1 + g2.round2) <
                    (g3.round1 + g3.round2))[0]   AS sqlrankR2
FROM g1 as g3
ORDER BY sqlrankR2

Results:
T ToPar player sqlrankR2
138 -6 "Bingjie"    1
140 -4 "Johan"      2
140 -4 "Sitaram"    2
143 -1 "Chang"      4
145 1 "Isha"       5
148 4 "Marco"      6
```

With window functions in Mad-Hatter (upcoming release)

This query returns the player, the total after two rounds (T), how much of the score is over/under par (ToPar), and then **ranks** them based on the scores of the first two rounds. This is the NEW functionality in Mad-Hatter. The time complexity of this is O(N), meaning execution time will only increase linearly.

```
SELECT    player                              AS player,
          (round1+round2)                     AS T,
          ((round1+round2) - 144)             AS ToPar,
          RANK() OVER(ORDER BY (round1+round2)) AS rankR2
FROM golf;



T ToPar player    rankR2
138 -6 "Bingjie"    1
140 -4 "Johan"      2
140 -4 "Sitaram"    2
143 -1 "Chang"      4
145 1 "Isha"      5
148 4 "Ma
```


Observ

The query exp d clearly.

1. Perform scenario is much better. We plan to measure.
2. When the ranking requirements depend on multiple documents, the query becomes quite complex to write, optimize, and run

2. When the ranking requirements depend on multiple documents, the query becomes quite complex to write, optimize, and run.

3. All of this affects the TCO overall.

Now, let's create an expanded dashboard.

Show add dense rank, row number, who's ahead, and the number of strokes behind the leader. All very common things in a reporting resituation. You're seeing the new window function whenever you see the OVER() clause. The query below has six window functions.

```
SELECT    player                                AS player,
          (round1+round2)                       AS T,
          ((round1+round2) - 144)               AS ToPar,
          RANK() OVER(ORDER BY (round1+round2)) AS rankR2,
          DENSE_RANK() OVER (ORDER BY (round1+round2)) AS rankR2Dense,
          ROW_NUMBER() OVER() rownum,
          ((round1+round2) -
              FIRST_VALUE(round1+round2)
                OVER(ORDER BY (round1+round2))) AS strokesbehind,
          RANK() OVER(ORDER BY (round1))        AS rankR1,
          LAG(player, 1, "None") OVER(ORDER BY round1+round2)
                                                AS inFront
FROM golf
ORDER BY rankR2


T ToPar inFront   player    rankR1  rankR2 rankR2Dense rownum strokesbehind
138 -6 "None"    "Bingjie" 3 1 1 3 0
140 -4 "Johan"   "Sitaram" 2 2 2 2 2
140 -4 "Bingjie" "Johan" 4 2 2 4 2
143 -1 "Sitaram" "Chang" 1 4 3 1 5
145 1 "Chang"  "Isha"        5 5 4 5 7
148 4 "Isha"   "Marco" 6 6 5 6 10
```

As you saw earlier, doing this query with **six window functions using** the subquery method will be a larger effort, expensive, error-prone query.

In addition to [illegible]UNT, SUM, AVG, etc) as window functions, Sitaram has added the following window functions. The syntax and semantics each of them a[illegible]d well-described in the articles of the reference section below.

- RANK()
- DENSE_R[illegible]
- PERCENT[illegible]
- CUME_DIST()

- CUME_DIST()
- NTILE()
- RATIO_TO_REPORT()
- ROW_NUMBER()
- LAG()
- FIRST_VALUE()
- LAST_VALUE()
- NTH_VALUE()
- LEAD()

References

1. Probably the Coolest SQL Feature: Window Functions
2. A Window into the World of Analytic Functions
3. Oracle Reference

Further Reading

- WINDOW Function
- How to Avoid Excessive Sorts in Window Functions

Like This Article? Read More From DZone

Parallelism: Estimated Costs of All Queries

Verifying Query Performance Using ProxySQL

Keeping MySQL in Check: Real-Life Advice to Tame Daily Performance Degradations

Free DZone Refcard
Working With Time Series Data