**[Add-Equals operator](http://am2.co/2016/05/add-equals-operator/)**

[Andy](http://www.ImpatientDBA.com)

[May 24, 2016](http://am2.co/2016/05/add-equals-operator/)

FROM: <http://am2.co/2016/05/add-equals-operator/>

**Shortcuts & tips from an Impatient DBA**

This series of short posts will demonstrate some lesser-known features, keyboard shortcuts, and other tips that make my day as a DBA more productive. None of these are groundbreaking super-secret features–they are the little things that I do as part of my daily work that make me more efficient.

**Concatenating strings**

You’ve written code that builds out a string. The most common reason I do it is when I’m writing dynamic SQL for my automation code. In fact, you’re probably already thinking of the code you wrote where you did something like this:

DECLARE @sql varchar(4000);  
SET @sql = ‘SELECT Col001 FROM [table] ‘;  
SET @sql = @sql + ‘ WHERE 1=2;’; — Concatenating @sql manually  
SELECT @sql;

I’ve been writing code like that for a decade….which is the problem. Ever since SQL Server 2008, I’ve been wasting precious keystrokes.

**There’s a better way**

SQL Server 2008 introduced an Add-Equals operator (+=) to T-SQL. Somehow, this hidden gem slipped past me. That whole “@sql = @sql +” nonsense can be made a lot easier to read.

DECLARE @sql varchar(4000);  
SET @sql = ‘SELECT Col001 FROM [table] ‘;  
SET @sql += ‘ WHERE 1=2;’; — Concatenation operator  
SELECT @sql;

This is logically equivalent to the first version of the code, but I find it makes for more readable code. It just looks cleaner.

For those of us who are looking to maximize efficiency, this could save a whole lot of key strokes.

**But wait, there’s more**

[If you call now, you can have not one, but TWO ShamWows](https://www.youtube.com/watch?v=QwRISkyV_B8)!

DECLARE @shamwow int = 0;  
SET @shamwow += 1;  
SET @shamwow += 1;  
SELECT @shamwow AS ShamWow;

Yup, **the Add-Equals operator** **works with both strings and numbers**.

So the next time you talk yourself into writing a loop or a cursor, and you implement a counter, you can save some keystrokes. That makes it efficient, right?

DECLARE @count int = 3;  
PRINT ‘Numbers less than 100 that are also multiples of three.’  
WHILE @count < 100  
BEGIN  
PRINT @Count;  
SET @Count += 3;  
END;