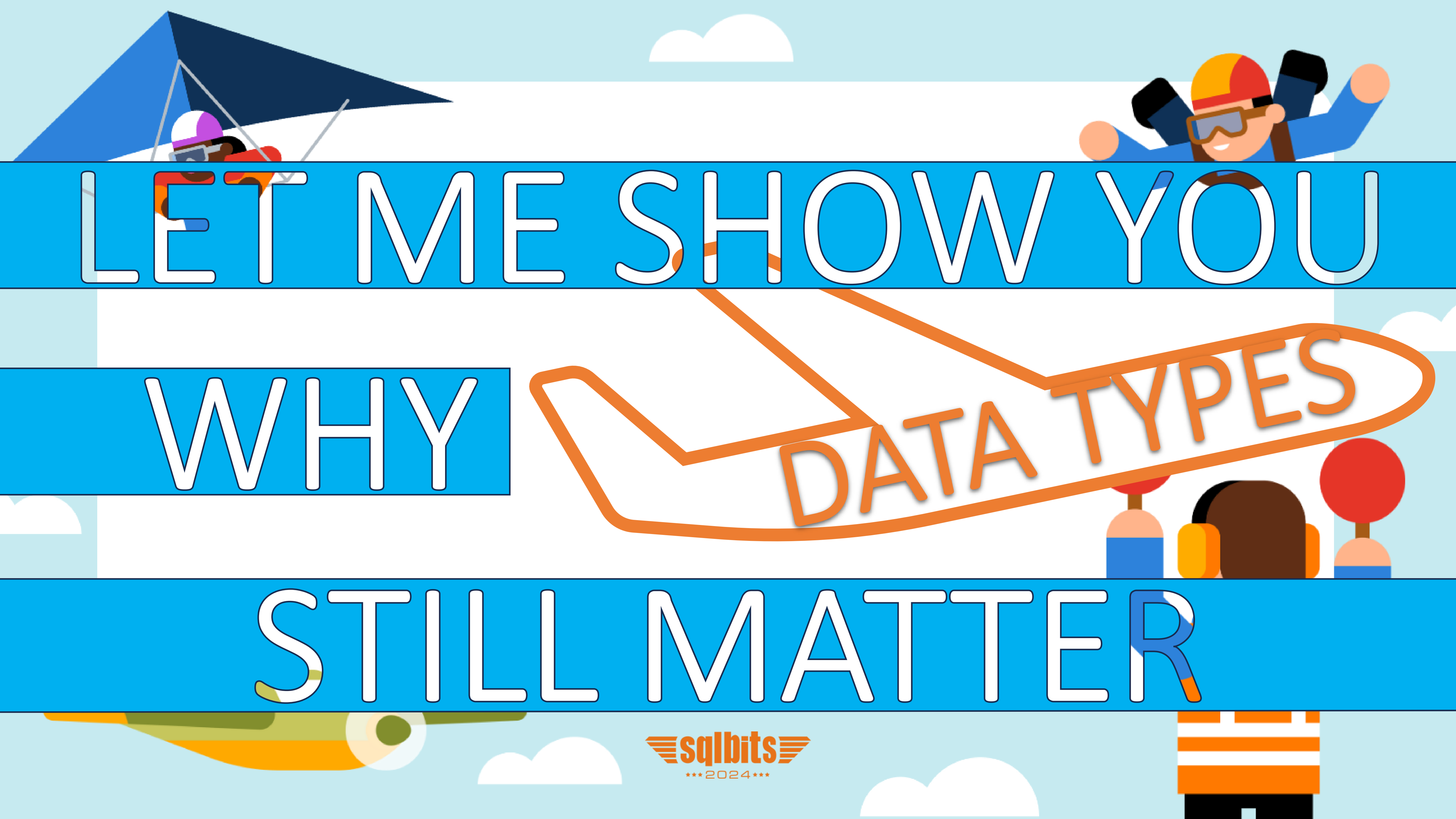


LET ME SHOW YOU WHY DATA TYPES STILL MATTER

sqlbits
2024

FEEDBACK

https://sqlb.it/?12523

sqlbits
2024

Who am I?

Cláudio Silva
(He / Him)

What do I do?

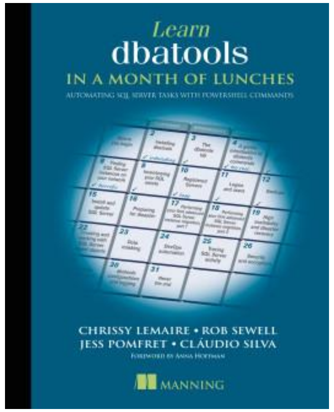
DATAmasterminds

- Performance Tuning
- Automation

Open-Source Contributor

- dbatools
- dbachecks

MVP
Microsoft®
Most Valuable
Professional



Learn
dbatools
IN A MONTH OF LUNCHES
AUTOMATING SQL SERVER TASKS WITH POWERSHELL COMMANDS

CHRISSY LEMAIRE • ROB SEWELL
JESS POMFRET • CLÁUDIO SILVA
FOREWORD BY ANNA HOFFMAN

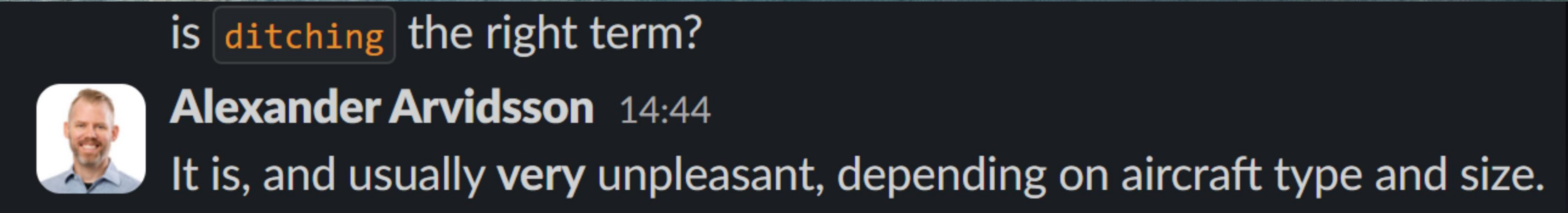
MANNING

in claudioessilva

@claudioessilva

claudioessilva.eu

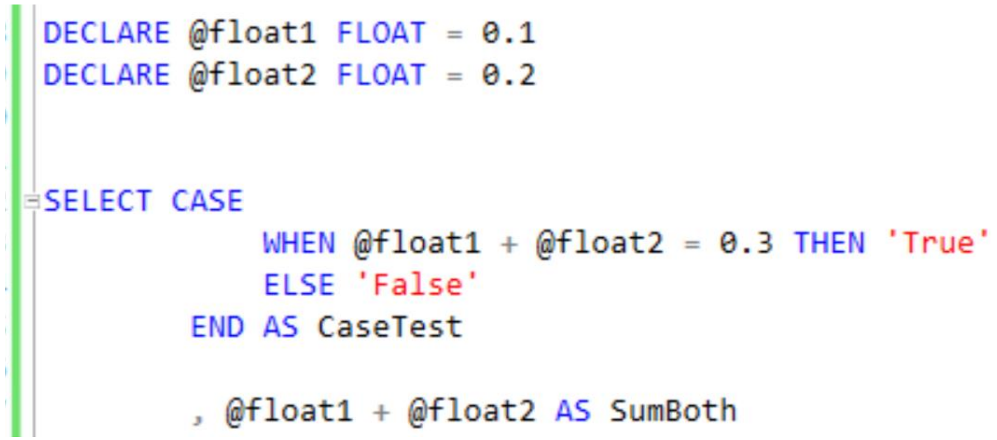
claudioessilva



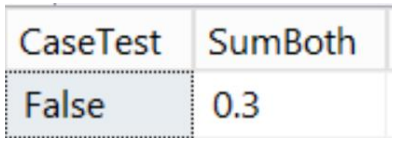
is `ditching` the right term?

Alexander Arvidsson 14:44

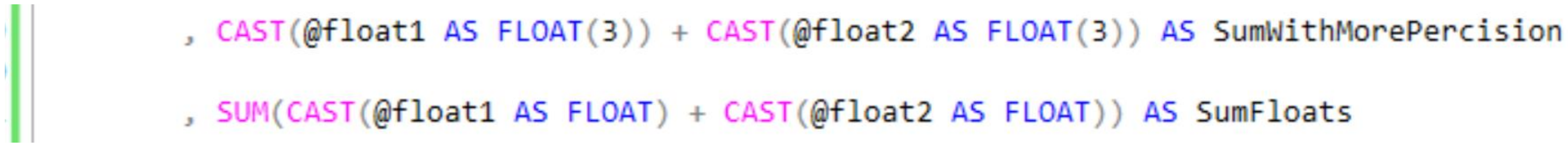
It is, and usually **very** unpleasant, depending on aircraft type and size.



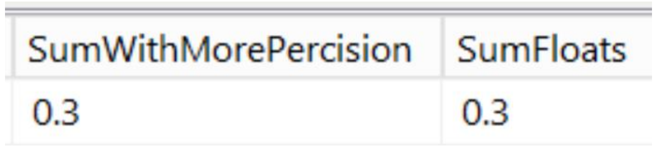
```sql
DECLARE @float1 FLOAT = 0.1
DECLARE @float2 FLOAT = 0.2


SELECT CASE
            WHEN @float1 + @float2 = 0.3 THEN 'True'
            ELSE 'False'
        END AS CaseTest

        , @float1 + @float2 AS SumBoth
```



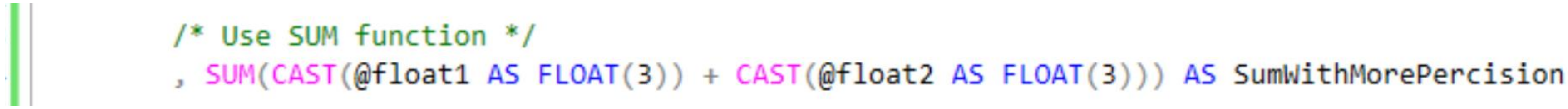
CaseTest	SumBoth
False	0.3



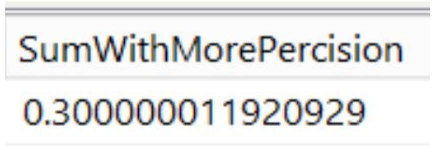
```
, CAST(@float1 AS FLOAT(3)) + CAST(@float2 AS FLOAT(3)) AS SumWithMorePercision

, SUM(CAST(@float1 AS FLOAT) + CAST(@float2 AS FLOAT)) AS SumFloats
```



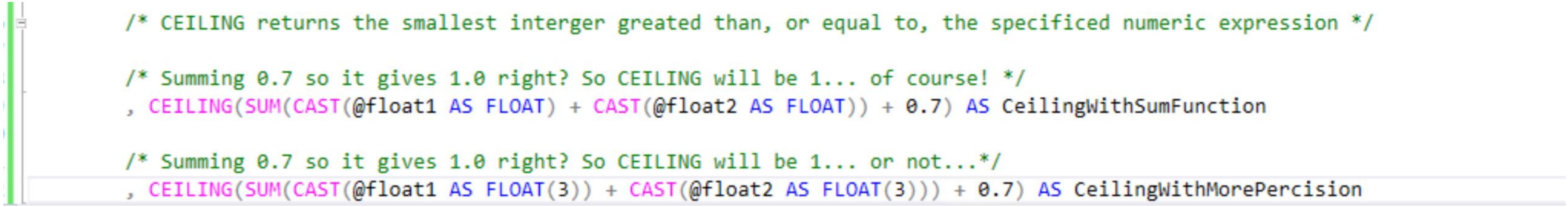
SumWithMorePercision	SumFloats
0.3	0.3



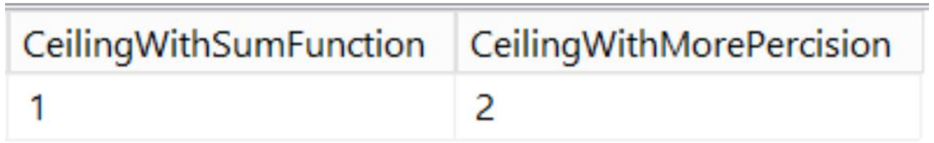
```
    /* Use SUM function */
    , SUM(CAST(@float1 AS FLOAT(3)) + CAST(@float2 AS FLOAT(3))) AS SumWithMorePercision
```



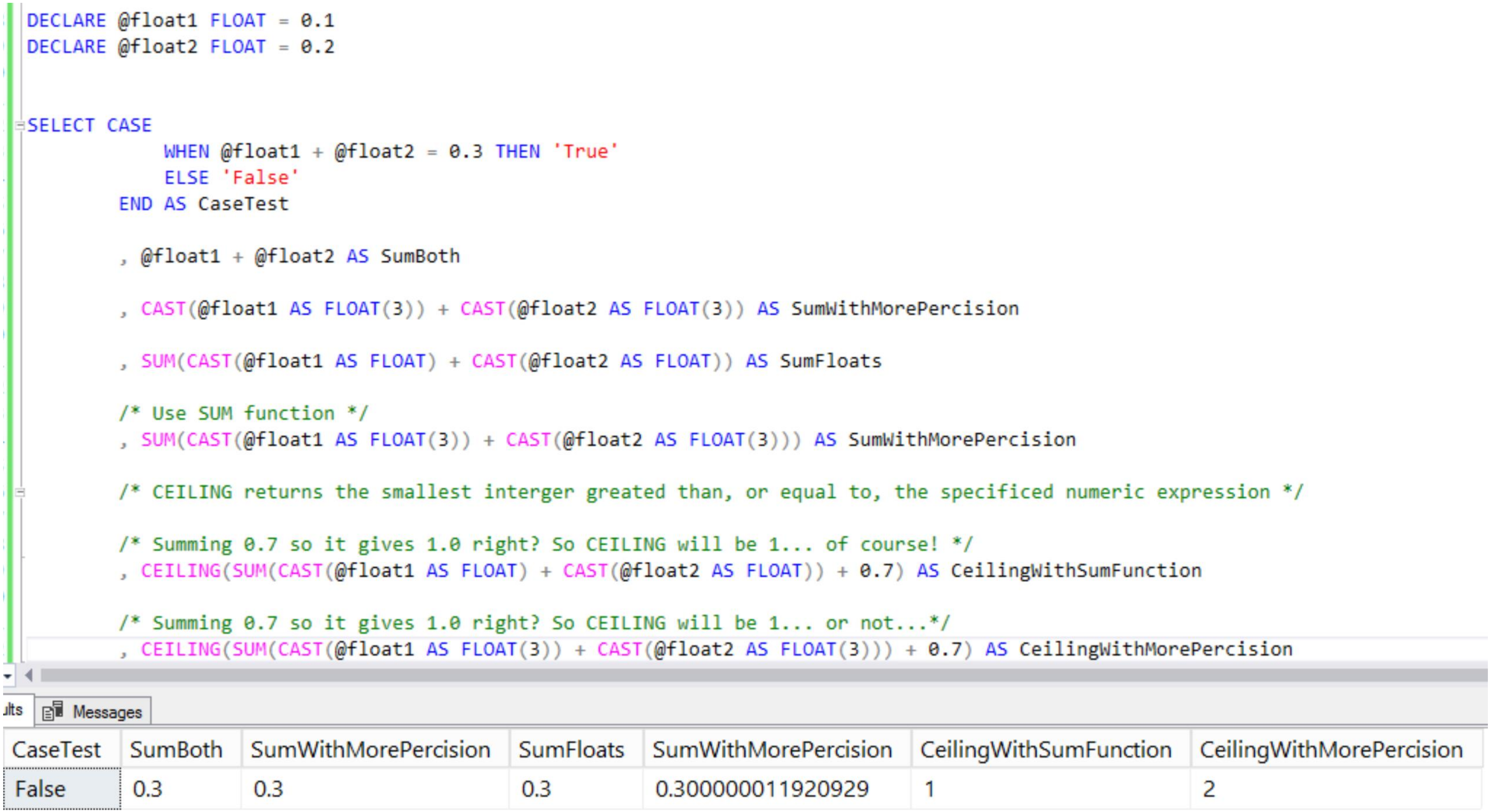
SumWithMorePercision
0.300000011920929



```
/* CEILING returns the smallest interger greated than, or equal to, the specificed numeric expression */

/* Summing 0.7 so it gives 1.0 right? So CEILING will be 1... of course! */
, CEILING(SUM(CAST(@float1 AS FLOAT) + CAST(@float2 AS FLOAT)) + 0.7) AS CeilingWithSumFunction

/* Summing 0.7 so it gives 1.0 right? So CEILING will be 1... or not...*/
, CEILING(SUM(CAST(@float1 AS FLOAT(3)) + CAST(@float2 AS FLOAT(3))) + 0.7) AS CeilingWithMorePercision
```



CeilingWithSumFunction	CeilingWithMorePercision
1	2



```sql
DECLARE @float1 FLOAT = 0.1
DECLARE @float2 FLOAT = 0.2


SELECT CASE
            WHEN @float1 + @float2 = 0.3 THEN 'True'
            ELSE 'False'
        END AS CaseTest

        , @float1 + @float2 AS SumBoth

        , CAST(@float1 AS FLOAT(3)) + CAST(@float2 AS FLOAT(3)) AS SumWithMorePercision

        , SUM(CAST(@float1 AS FLOAT) + CAST(@float2 AS FLOAT)) AS SumFloats

        /* Use SUM function */
        , SUM(CAST(@float1 AS FLOAT(3)) + CAST(@float2 AS FLOAT(3))) AS SumWithMorePercision

        /* CEILING returns the smallest interger greated than, or equal to, the specificed numeric expression */

        /* Summing 0.7 so it gives 1.0 right? So CEILING will be 1... of course! */
        , CEILING(SUM(CAST(@float1 AS FLOAT) + CAST(@float2 AS FLOAT)) + 0.7) AS CeilingWithSumFunction

        /* Summing 0.7 so it gives 1.0 right? So CEILING will be 1... or not...*/
        , CEILING(SUM(CAST(@float1 AS FLOAT(3)) + CAST(@float2 AS FLOAT(3))) + 0.7) AS CeilingWithMorePercision
```

CaseTest	SumBoth	SumWithMorePercision	SumFloats	SumWithMorePercision	CeilingWithSumFunction	CeilingWithMorePercision
False	0.3	0.3	0.3	0.300000011920929	1	2

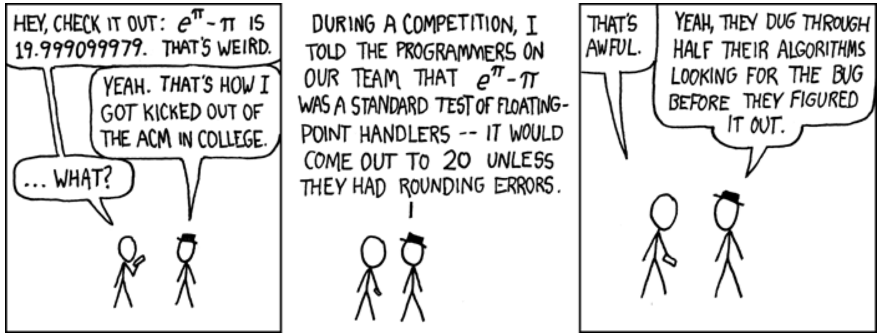
TAKEAWAY

Ok to use: For things you measure and don’t need lots of precision

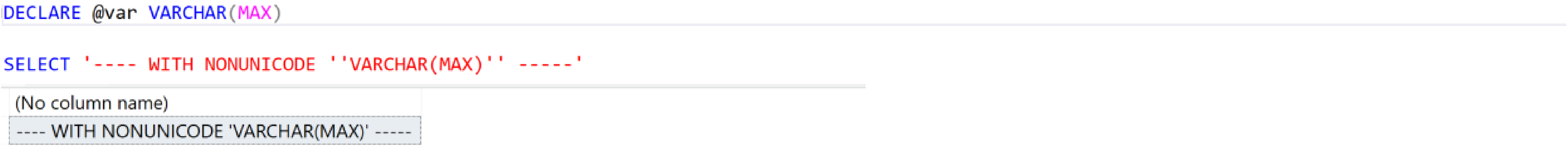
Don’t use: For things you count

[Floating Point Math (30000000000000004.com) - https://0.30000000000000004.com/](https://0.30000000000000004.com/)

[The Floating-Point Guide - What Every Programmer Should Know About Floating-Point Arithmetic - https://floating-point-gui.de/](https://floating-point-gui.de/)

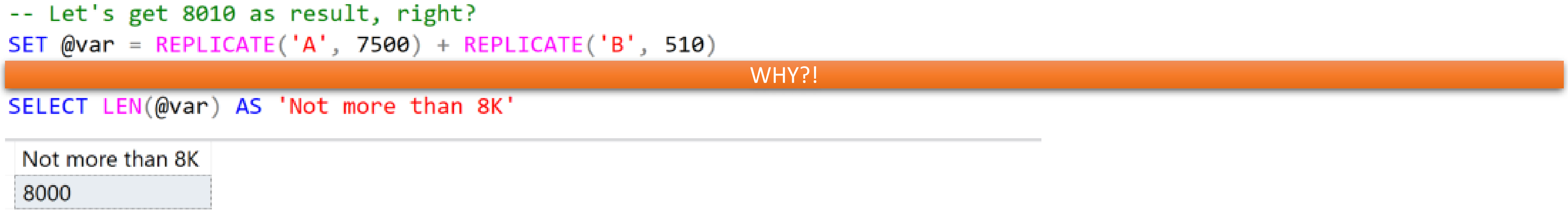

HEY, CHECK IT OUT: $e^{\pi} - \pi$ IS 19.999099979. THAT'S WEIRD.
YEAH. THAT'S HOW I GOT KICKED OUT OF THE ACM IN COLLEGE.
... WHAT?
DURING A COMPETITION, I TOLD THE PROGRAMMERS ON OUR TEAM THAT $e^{\pi} - \pi$ WAS A STANDARD TEST OF FLOATING-POINT HANDLERS -- IT WOULD COME OUT TO 20 UNLESS THEY HAD ROUNDING ERRORS.
THAT'S AWFUL.
YEAH, THEY DUG THROUGH HALF THEIR ALGORITHMS LOOKING FOR THE BUG BEFORE THEY FIGURED IT OUT.

(n) varchar(max)



```
DECLARE @var VARCHAR(MAX)

SELECT '---- WITH NONUNICODE ''VARCHAR(MAX)'' -----'
```

```
(No column name)
---- WITH NONUNICODE 'VARCHAR(MAX)' -----
```

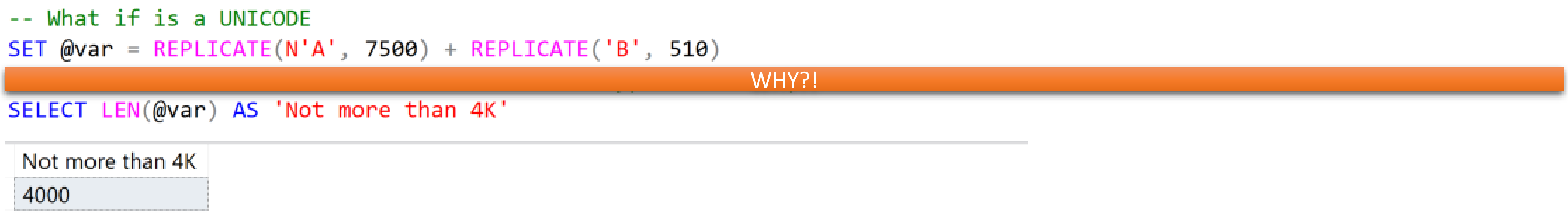


```sql
-- Let's get 8010 as result, right?
SET @var = REPLICATE('A', 7500) + REPLICATE('B', 510)
```

WHY?!

```sql
SELECT LEN(@var) AS 'Not more than 8K'
```

```
Not more than 8K
8000
```

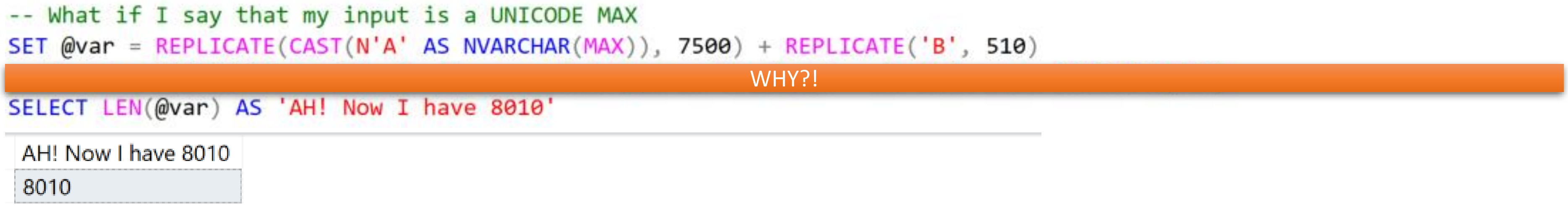


```
-- What if is a UNICODE
SET @var = REPLICATE(N'A', 7500) + REPLICATE('B', 510)
```

WHY?!

```
SELECT LEN(@var) AS 'Not more than 4K'
```

```
Not more than 4K
4000
```



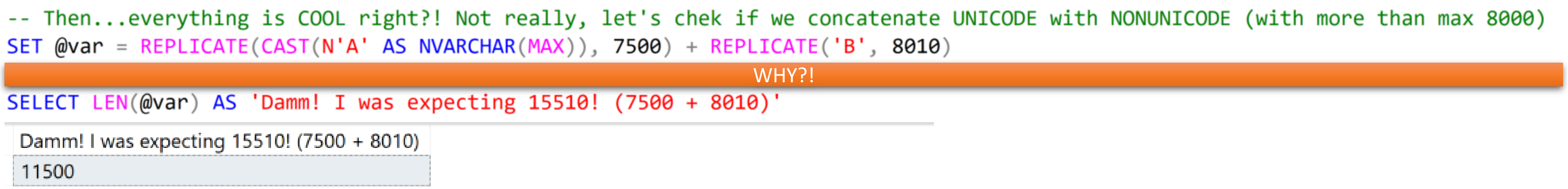
```
-- What if I say that my input is a UNICODE MAX
SET @var = REPLICATE(CAST(N'A' AS NVARCHAR(MAX)), 7500) + REPLICATE('B', 510)
```

WHY?!

```
SELECT LEN(@var) AS 'AH! Now I have 8010'
```

```
AH! Now I have 8010
8010
```

Here lies the “problem”

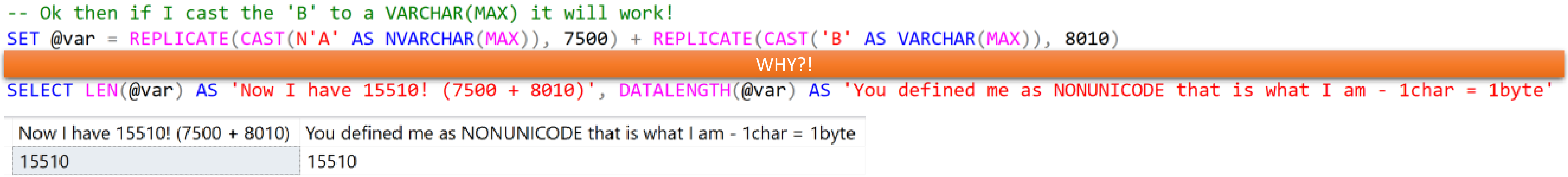


```
-- Then...everything is COOL right?! Not really, let's chek if we concatenate UNICODE with NONUNICODE (with more than max 8000)
SET @var = REPLICATE(CAST(N'A' AS NVARCHAR(MAX)), 7500) + REPLICATE('B', 8010)
```

WHY?!

```
SELECT LEN(@var) AS 'Damm! I was expecting 15510! (7500 + 8010)'
```

```
Damm! I was expecting 15510! (7500 + 8010)
11500
```

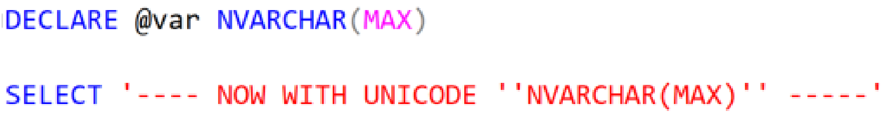


```sql
-- Ok then if I cast the 'B' to a VARCHAR(MAX) it will work!
SET @var = REPLICATE(CAST(N'A' AS NVARCHAR(MAX)), 7500) + REPLICATE(CAST('B' AS VARCHAR(MAX)), 8010)
```

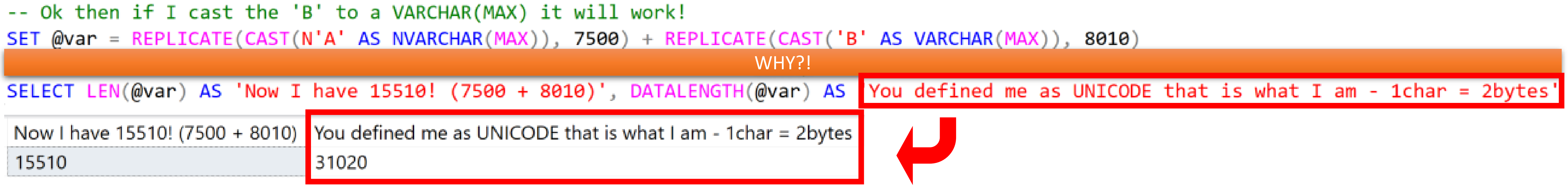
WHY?!

```sql
SELECT LEN(@var) AS 'Now I have 15510! (7500 + 8010)', DATALENGTH(@var) AS 'You defined me as NONUNICODE that is what I am - 1char = 1byte'
```

```
Now I have 15510! (7500 + 8010)   You defined me as NONUNICODE that is what I am - 1char = 1byte
15510                             15510
```



```
DECLARE @var NVARCHAR(MAX)

SELECT '---- NOW WITH UNICODE ''NVARCHAR(MAX)'' -----'
```



```
-- Ok then if I cast the 'B' to a VARCHAR(MAX) it will work!
SET @var = REPLICATE(CAST(N'A' AS NVARCHAR(MAX)), 7500) + REPLICATE(CAST('B' AS VARCHAR(MAX)), 8010)
```

WHY?!

```
SELECT LEN(@var) AS 'Now I have 15510! (7500 + 8010)', DATALENGTH(@var) AS 'You defined me as UNICODE that is what I am - 1char = 2bytes'
```

```
Now I have 15510! (7500 + 8010)   You defined me as UNICODE that is what I am - 1char = 2bytes
15510                             31020
```

2 Columns

- ID – BIGINT – 8 bytes
- GUID (just the name 😆) – NVARCHAR(50) – 100 bytes (max)

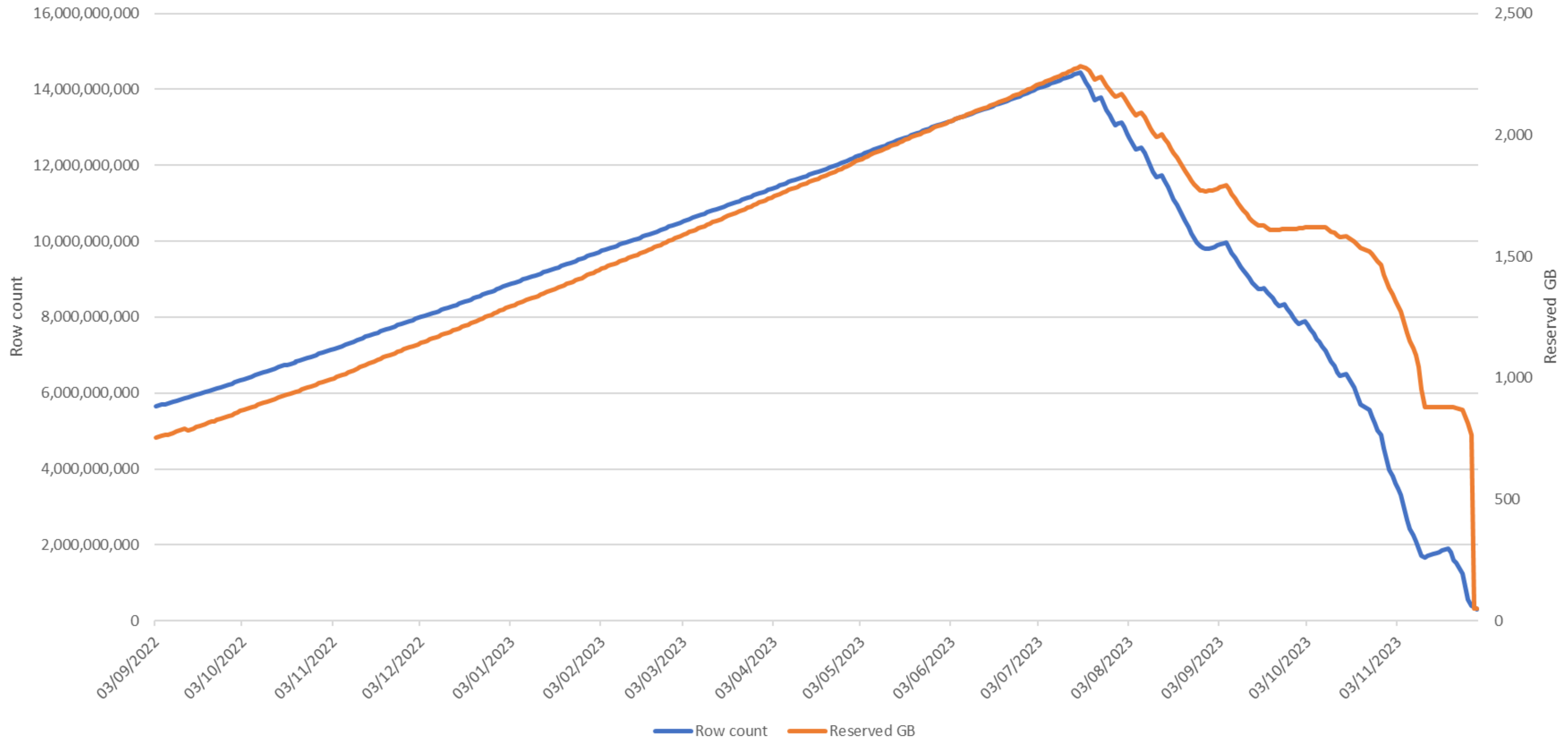
Let’s say that:

- Each record, on average would be 80 bytes (8 bytes + 72 bytes)
    - 1M -> 80 MB;
    - 1Bi -> 80000 MB -> 80 GB
    - 14Bi -> 1.12 TB

The MAX(LEN(GUID)) = 36 - With VARCHAR(36)

- Each record, on average would be 44 bytes (8 bytes + 36 bytes)
    - 1M -> 44 MB
    - 1Bi -> 44000 MB -> 44 GB
    - 14Bi -> 616 GB

The graph



sqlbits
★★★2024★★★

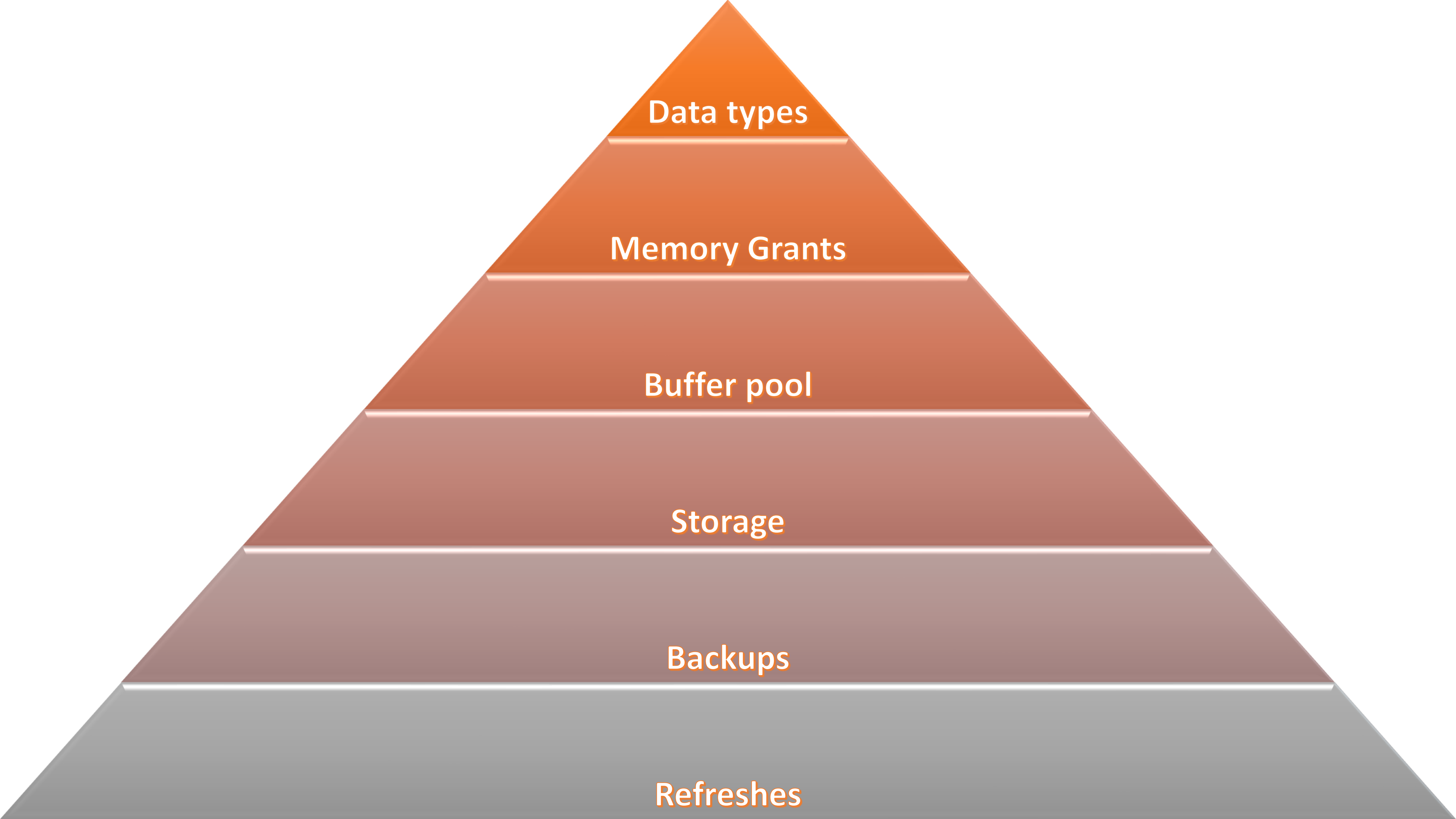
TAKEAWAY

On concatenations:

- Be aware of truncations!
- Try to use the same type (UNICODE/NONUNICODE)!

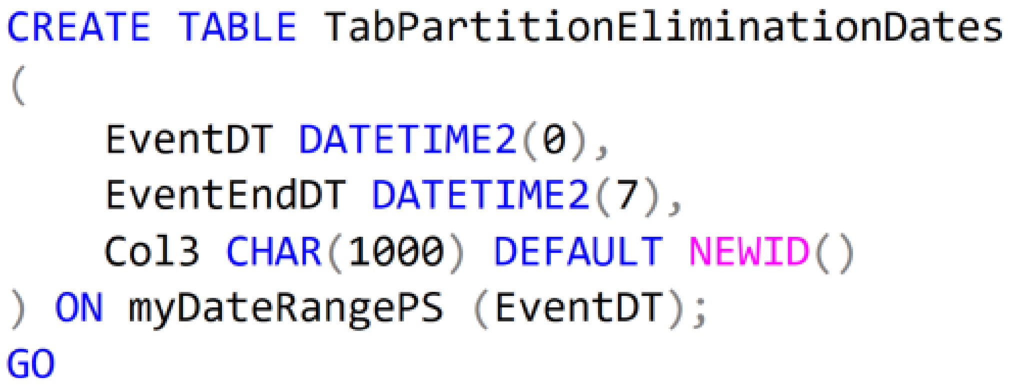
Tables

- Choose the type and size wisely



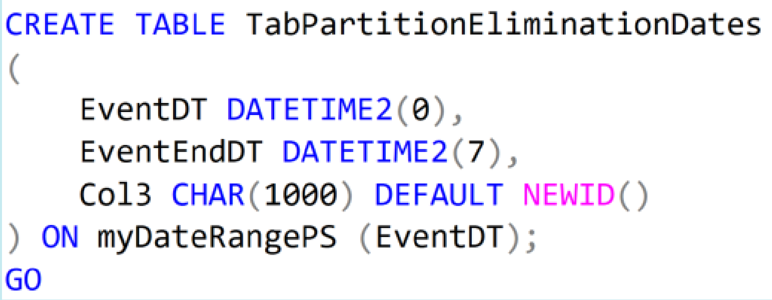
Data types
Memory Grants
Buffer pool
Storage
Backups
Refreshes

Partition elimination – Isn't happening

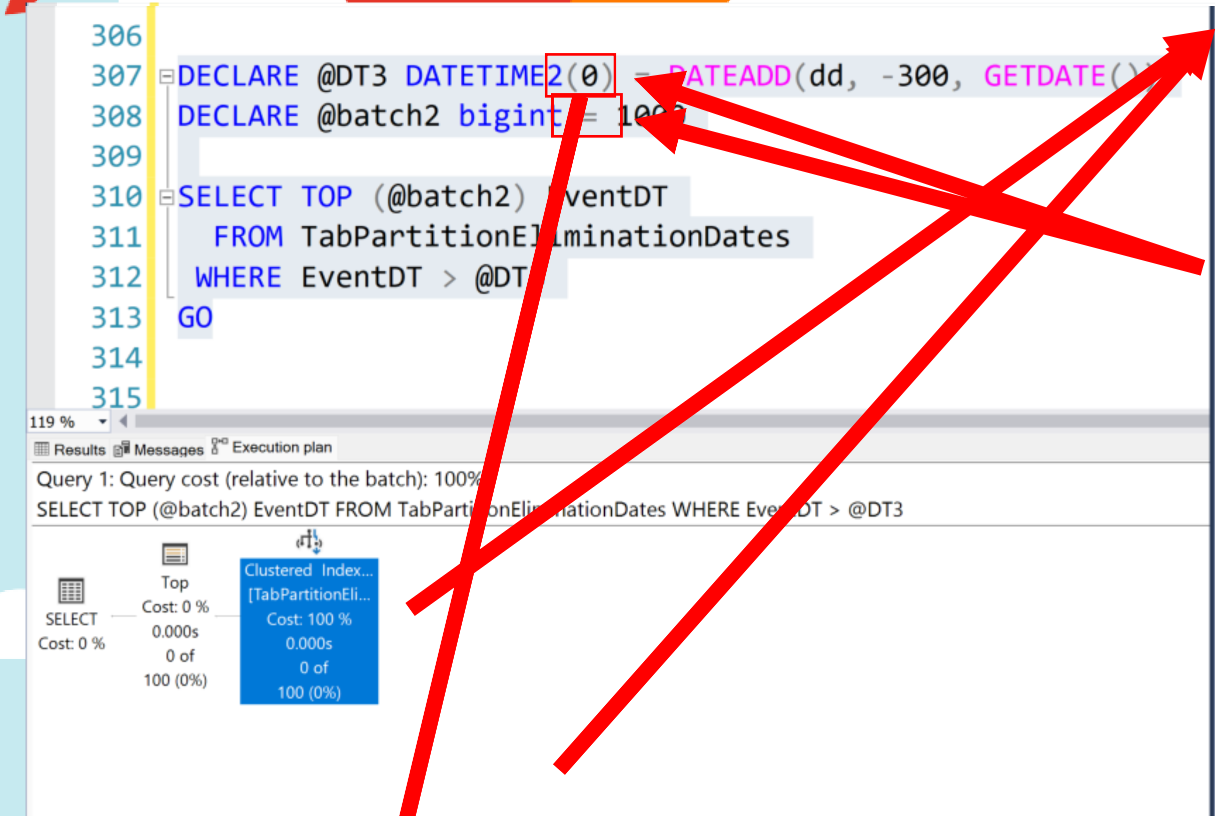


```
CREATE TABLE TabPartitionEliminationDates
(
    EventDT DATETIME2(0),
    EventEndDT DATETIME2(7),
    Col3 CHAR(1000) DEFAULT NEWID()
) ON myDateRangePS (EventDT);
GO
```

Partition elimination – Isn’t happening

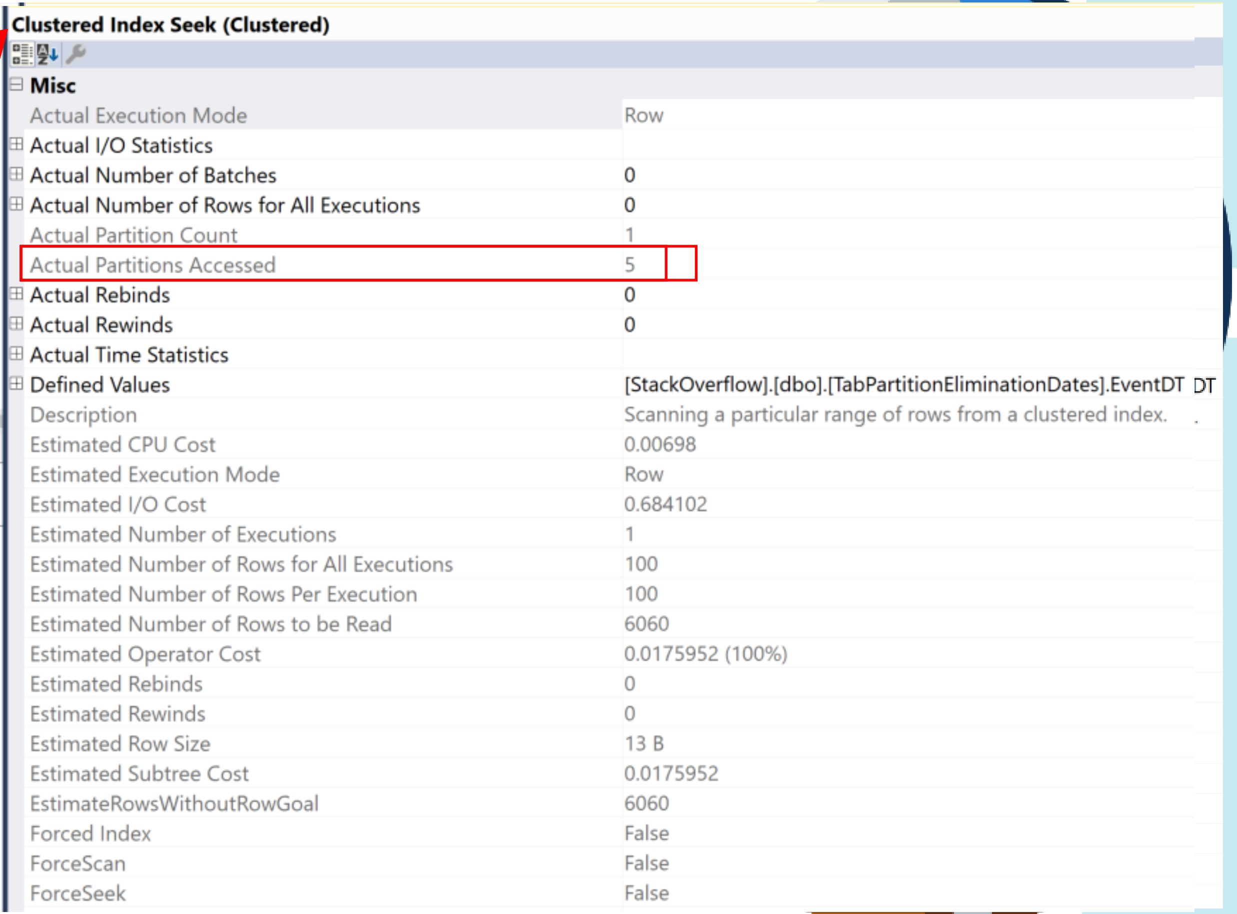


```
CREATE TABLE TabPartitionEliminationDates
(
    EventDT DATETIME2(0),
    EventEndDT DATETIME2(7),
    Col3 CHAR(1000) DEFAULT NEWID()
) ON myDateRangePS (EventDT);
GO
```



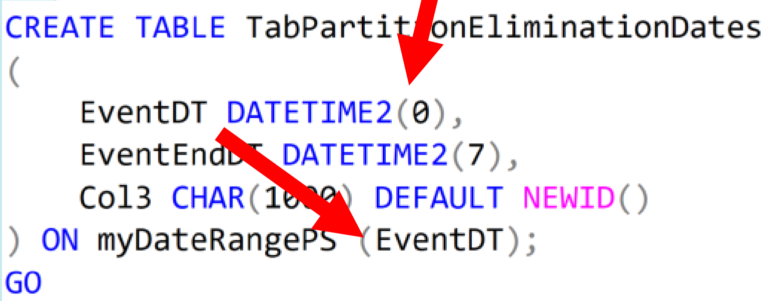
```sql
DECLARE @DT3 DATETIME2(0) = DATEADD(dd, -300, GETDATE())
DECLARE @batch2 bigint = 1000

SELECT TOP (@batch2) EventDT
  FROM TabPartitionEliminationDates
 WHERE EventDT > @DT3
GO
```

Query 1: Query cost (relative to the batch): 100%
SELECT TOP (@batch2) EventDT FROM TabPartitionEliminationDates WHERE EventDT > @DT3



Clustered Index Seek (Clustered)

Misc	
Actual Execution Mode	Row
Actual I/O Statistics	
Actual Number of Batches	0
Actual Number of Rows for All Executions	0
Actual Partition Count	1
Actual Partitions Accessed	5
Actual Rebinds	0
Actual Rewinds	0
Actual Time Statistics	
Defined Values	[StackOverflow].[dbo].[TabPartitionEliminationDates].EventDT DT
Description	Scanning a particular range of rows from a clustered index.
Estimated CPU Cost	0.00698
Estimated Execution Mode	Row
Estimated I/O Cost	0.684102
Estimated Number of Executions	1
Estimated Number of Rows for All Executions	100
Estimated Number of Rows Per Execution	100
Estimated Number of Rows to be Read	6060
Estimated Operator Cost	0.0175952 (100%)
Estimated Rebinds	0
Estimated Rewinds	0
Estimated Row Size	13 B
Estimated Subtree Cost	0.0175952
EstimateRowsWithoutRowGoal	6060
Forced Index	False
ForceScan	False
ForceSeek	False



```sql
CREATE TABLE TabPartitionEliminationDates
(
    EventDT DATETIME2(0),
    EventEndDT DATETIME2(7),
    Col3 CHAR(1000) DEFAULT NEWID()
) ON myDateRangePS (EventDT);
GO
```

sqlbits 2024

More examples

- Memory Grants
    - (n)varchar with and without MAX
    - sorts
    - RESOURCE_SEMAPHORE (wait type)
- Implicit convertions
    - Can lead to INDEX SCAN

Questions? Find me around

claudioessilva

@claudioessilva

claudioessilva.eu

claudioessilva

https://sqlb.it/?12523

sqlbits

2024