# SQL Server Performance

## Best Practices and Resources

**Mohit Gupta - SQLDBA**

# Agenda

- Overview
- Different aspects of SQL Server Performance
- SQL Server\Instance best practices
- SQL Database best practices
- Index\Statistics best practices
- Query Tuning best practices
- Types of Performance Issues
- Quick identification of the issue
- Useful open source tools\scripts
- Overview of the  sp_WhoisActive & First Responder Kit

# Overview

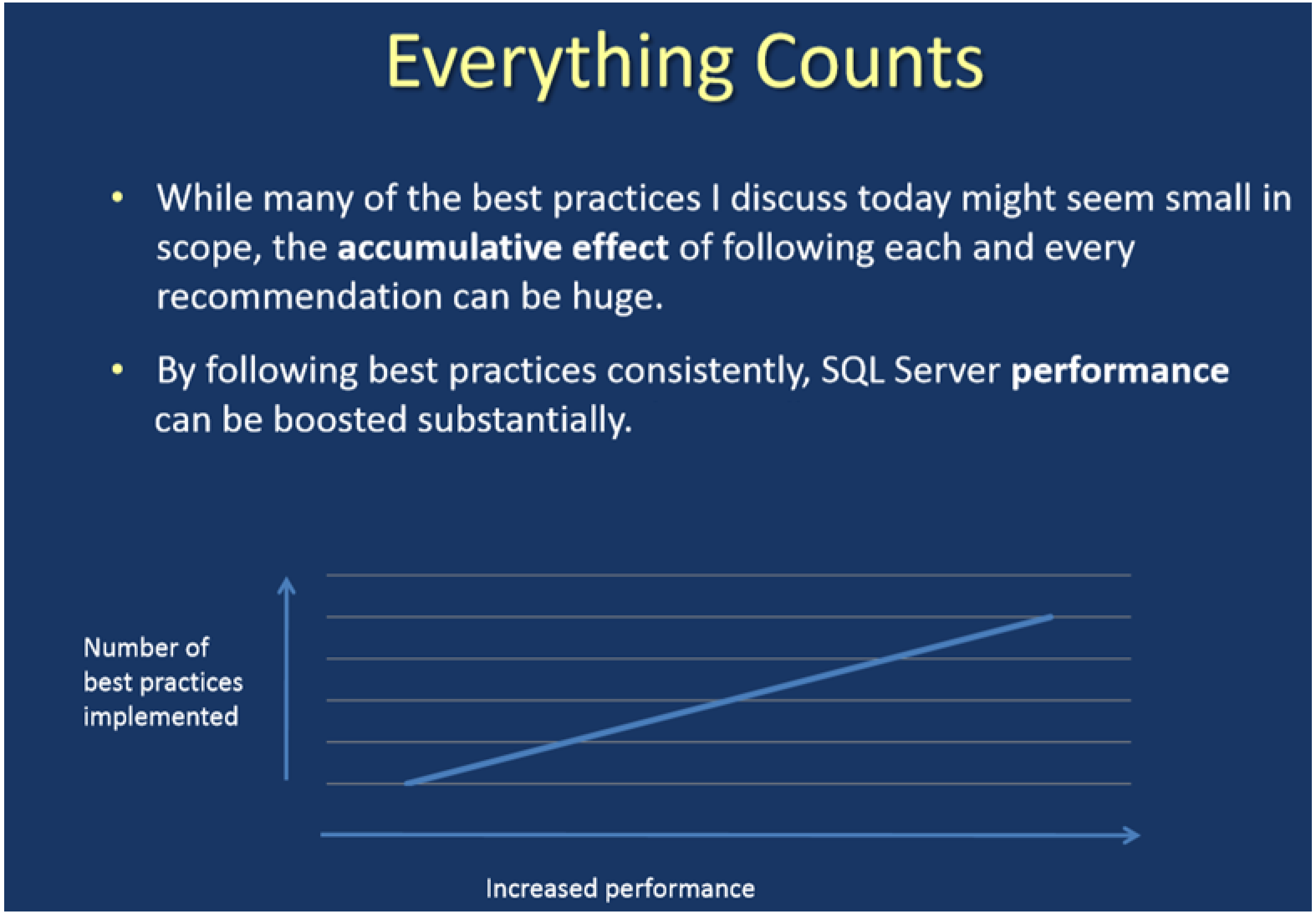
Users and Clients expectations are high with respect to SQL Server’s performance.

They want all the servers to perform like this for their queries and workload !!!!

But the servers might be performing like this !!!   So we need to tune it as much as we can …..

# Different aspects of SQL Server Performance tuning

- Server \ Instance level tuning
- Databases level tuning
- Indexes & Statistics tuning
- Query Tuning



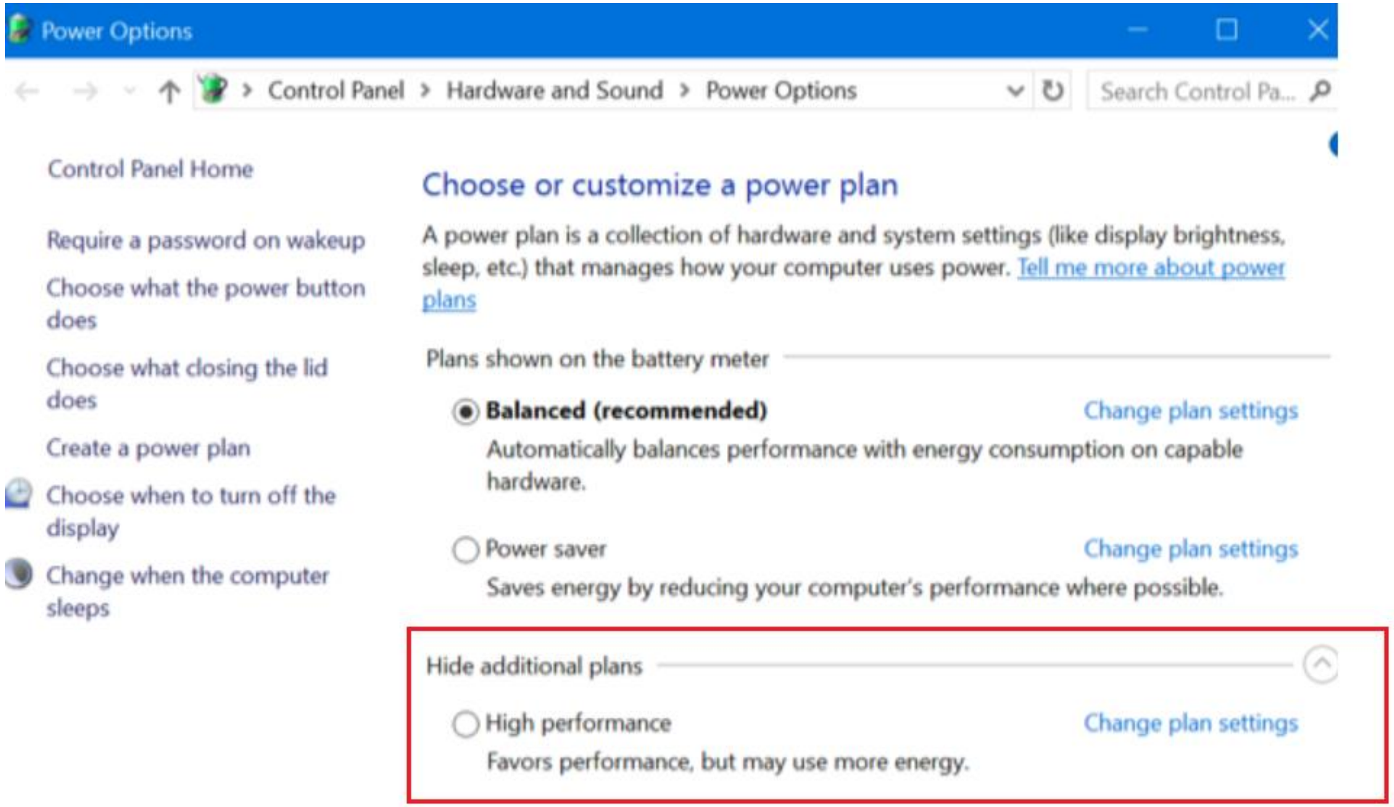
# Everything Counts

- While many of the best practices I discuss today might seem small in scope, the **accumulative effect** of following each and every recommendation can be huge.
- By following best practices consistently, SQL Server **performance** can be boosted substantially.

Number of best practices implemented
Increased performance

# SQL Server \ Instance level best practices

- Dedicated single instance as far as possible on a box. If virtualized, then VM for a single instance instead of multiple on one VM.
- Separate drives for Binaries, system databases, user database data files, log files, tempdb etc.
- Preferably SSD At least RAID 10 if not SSD. RAID 5 last option.
- Don't install or disable SQL Server services which are not required like SSIS , Full text, Browser etc.
- **Enable Instant file initialization**, by adding the service account to the Perform Volume Maintenance Tasks security policy. It speeds up CREATE DATABASE, ALTER DATABASE, RESTORE DATABASE, Auto growth. LDF files are not affected by this as Log files are always zero-initialized. Also TDE databases are not impacted.
- **Use the "Adjust for Best Performance" for the OS advanced system property** for the SQL service account profile if possible.
- **In the OS power setting "Power Options" , always select the power plan "High Performance " . May not help on VM.**
- 64k allocation for the NTFS disk or the mount volume for the SQL server instance. Format the drives with 64K allocation blocks
- **MAXDOP = 4 to start with for a multicore system , not more than 8 ( in DW )** which starts reverse impacting. Can refer MS recommendation based on NUMA but 4 is a good number. Always use an even number.
- **CTOP = At least 50, not the default 5 which is too low and misleading for optimizer.**



Power Options
Control Panel > Hardware and Sound > Power Options
Search Control Pa...
Control Panel Home
Require a password on wakeup
Choose what the power button does
Choose what closing the lid does
Create a power plan
Choose when to turn off the display
Change when the computer sleeps

Choose or customize a power plan

A power plan is a collection of hardware and system settings (like display brightness, sleep, etc.) that manages how your computer uses power. Tell me more about power plans

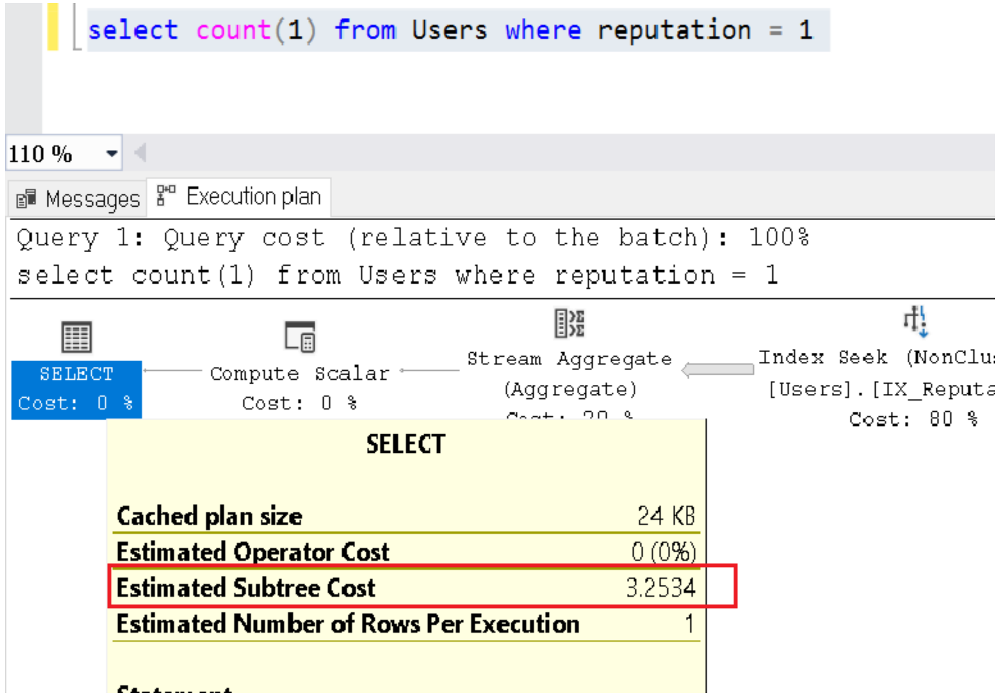
Plans shown on the battery meter

● **Balanced (recommended)** Change plan settings
Automatically balances performance with energy consumption on capable hardware.

○ Power saver Change plan settings
Saves energy by reducing your computer's performance where possible.

Hide additional plans

○ High performance Change plan settings
Favors performance, but may use more energy.



```
select count(1) from Users where reputation = 1
```

```
Query 1: Query cost (relative to the batch): 100%
select count(1) from Users where reputation = 1
```

**SELECT**

| | |
|---|---|
| **Cached plan size** | 24 KB |
| **Estimated Operator Cost** | 0 (0%) |
| **Estimated Subtree Cost** | 3.2534 |
| **Estimated Number of Rows Per Execution** | 1 |

- Max memory in such a way that you always have 4 GB or 10 % of RAM ( whichever is more ) for OS .
- **Trace 1117 & 1118** for avoiding SGAM page contention in tempdb. Not required SQL 2016 onwards.
- Enable the ”**Optimize for Ad hoc Workloads ”** option in the SQL Server instance settings., if your adhoc plan cache is 20-30% of total Plan Cache. 3rd party apps like DAX, CRM, SharePoint etc may be beneficial.
- Disable **priority boost** if it is enabled unintentionally.
- **DAC** should always be enabled for emergency.
- If using >= SQL 2016 instance and the DB compatibility level is also the latest , then Query optimizer fixes are effective already like latest cardinality estimator algorithms.
- If using >=SQL 2016 instance but the DB compatibility level is < 130 then make Query Optimizer Fixes = ON . SQL earlier versions: Enable trace flag **4199** at service level
- Configure the proper recommended size Page File for Virtual Memory.
- Ideally, don’t run antivirus locally. If local, exclude MDF, NDF, LDF, BAK, TRN and BCP files.

# First step done !!!

# SQL Databases level best practices

- When creating new MDFs and LDFs, pre-size them to eliminate/minimize auto growth events. Auto growth should NOT be % . 1024 MB for Data, 512 MB for Log is a good practice.
- Auto Create Statistics: On • Auto Update Stats: On • Auto Update Stats Async : On • Auto Shrink: Off • Auto Close : Off
- Page Verify: Use Checksum , don’t turn off. Will help in involving overhead of checksum calculation.
- If multiple data files, try to put in different drives.
- Always create a secondary data file and make that default. Always keep free space in Primary for metadata update.
- **VLF control.** Don’t allow more than 200. Log backups and proper auto growth helps in controlling. If log files are fragmented with high VLF you will see a lot of WRITELOG wait types
- Compatibility Level: Should be set to match current server version, unless there are compatibility problems -- 2016 - Cardinality estimator changed. **2019 – how the table variable be handled changed.**
- Enable Read Committed Snapshot (**RCSI**) mode for R\W heavy workload . Need to monitor tempdb usage.

- TEMPDB data files # For <= 8 cores = # of cores . For > 8 cores , 8 Tempdb data files . Not more than 8 advised.
- **NEVER keep a single TEMPDB file of course !!!** 2016 onwards, it advises during the install. Always try to size same and pre-grow the files to avoid auto growth.
- Don’t Shrink Files . If done manually , rebuild the indexes after the shrink is complete or at least update the statistics.
- **Quick rollback of heavy DML queries for VLDB , use ADR** ( Accelerated Database Recovery ) since the rollback of a DML is always single threaded.
- **Always enable the Query Store** ( supported versions ) with enough space for growth in Primary file.
- Use Data compression (row\page) if feasible to improve IO performance and effective use of memory, but the trade off for CPU should be considered and well tested thoroughly.

So far,so good
So far,so good

# Indexes \ Statistics best practices

- Missing, Unused, Duplicate indexes should be analyzed and handled regularly.
- Avoid using the native maintenance plans. Use 3rd party\open source solutions like OLA scripts.
- **Aggressive & Smart strategy** instead of full , for index and stats maintenance , specially for VLDB.

https://ola.hallengren.com/sql-server-index-and-statistics-maintenance.html

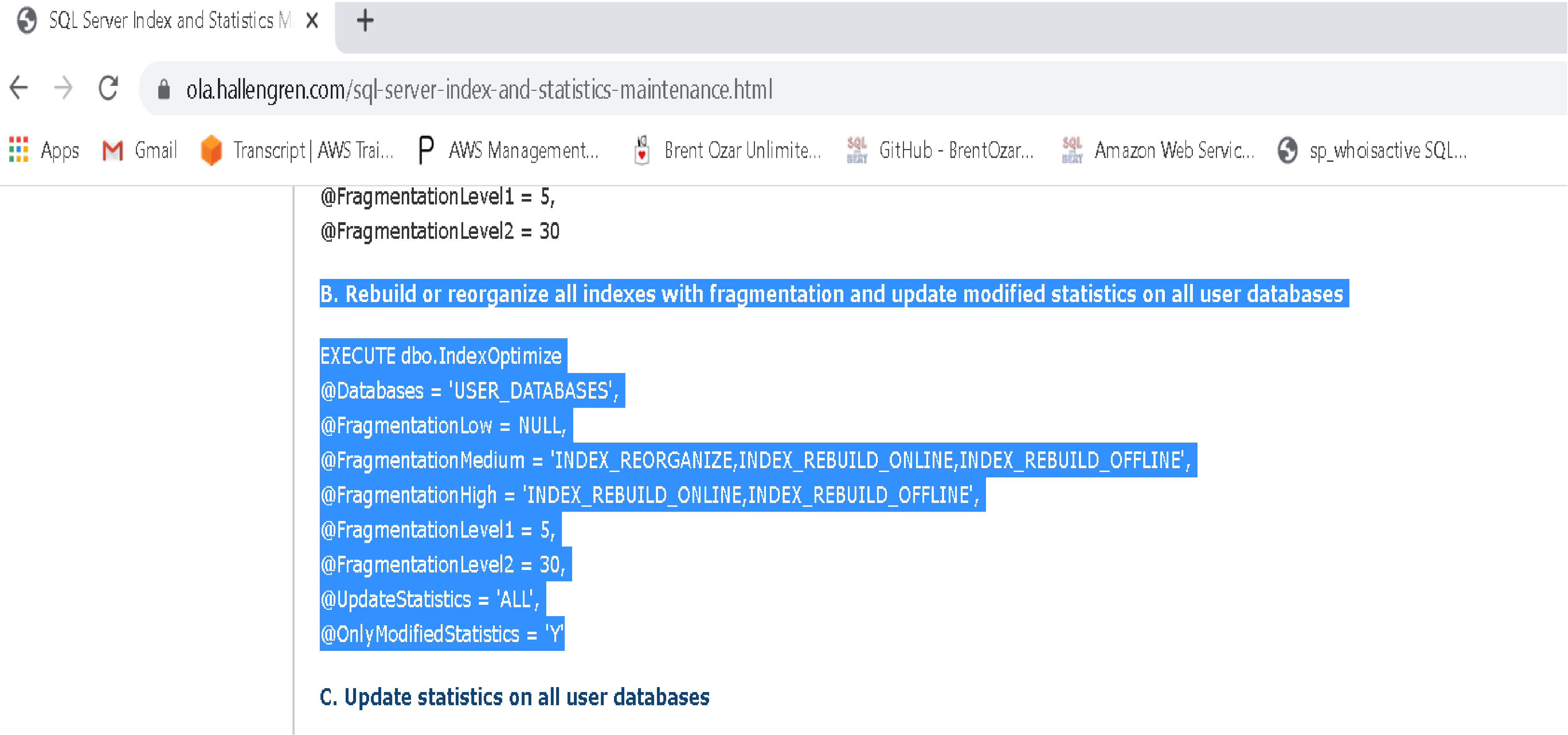
– If fragmentation is less <30%, then leave alone.

– If fragmentation is >30% and <70%, consider Reorganize & Update stats .

– If fragmentation >70%, consider Rebuild.

- Don’t simply update **all** statistics daily. Update only genuine modified ones.
- **Fill factor & Pad index - Keep pages 100 % filled** , unless you know why not to . The more you leave space, you are increasing DBCC , Backup, Physical read time and pressurizing buffer

- Avoid over-Indexing a Table - When a table has too many indexes, write operations become slower . Space usage is more. Check unused indexes.
- Under-Indexing a Table - An under-indexed table does not serve read queries effectively. Check missing indexes.
- Create clustered indexes -- will almost always perform better than heaps and will provide the necessary infrastructure to add non-clustered indexes efficiently when needed.
- Create primary key -- provides valuable information to the query optimizer that helps it make smart decisions when creating execution plans.
- In case of replication , don’t keep unwanted indexes on subscriber same as publisher. It will cause latency.
- Trace flag 2371 is on by default SQL 2016 onwards , else enable it on prior versions. Ensures dynamic updating statistics. Else , statistics would update when 20% of a table changed as before.
- **NC indexes fragmentation , doesn’t matter much as long as your Statistics are up to date !**



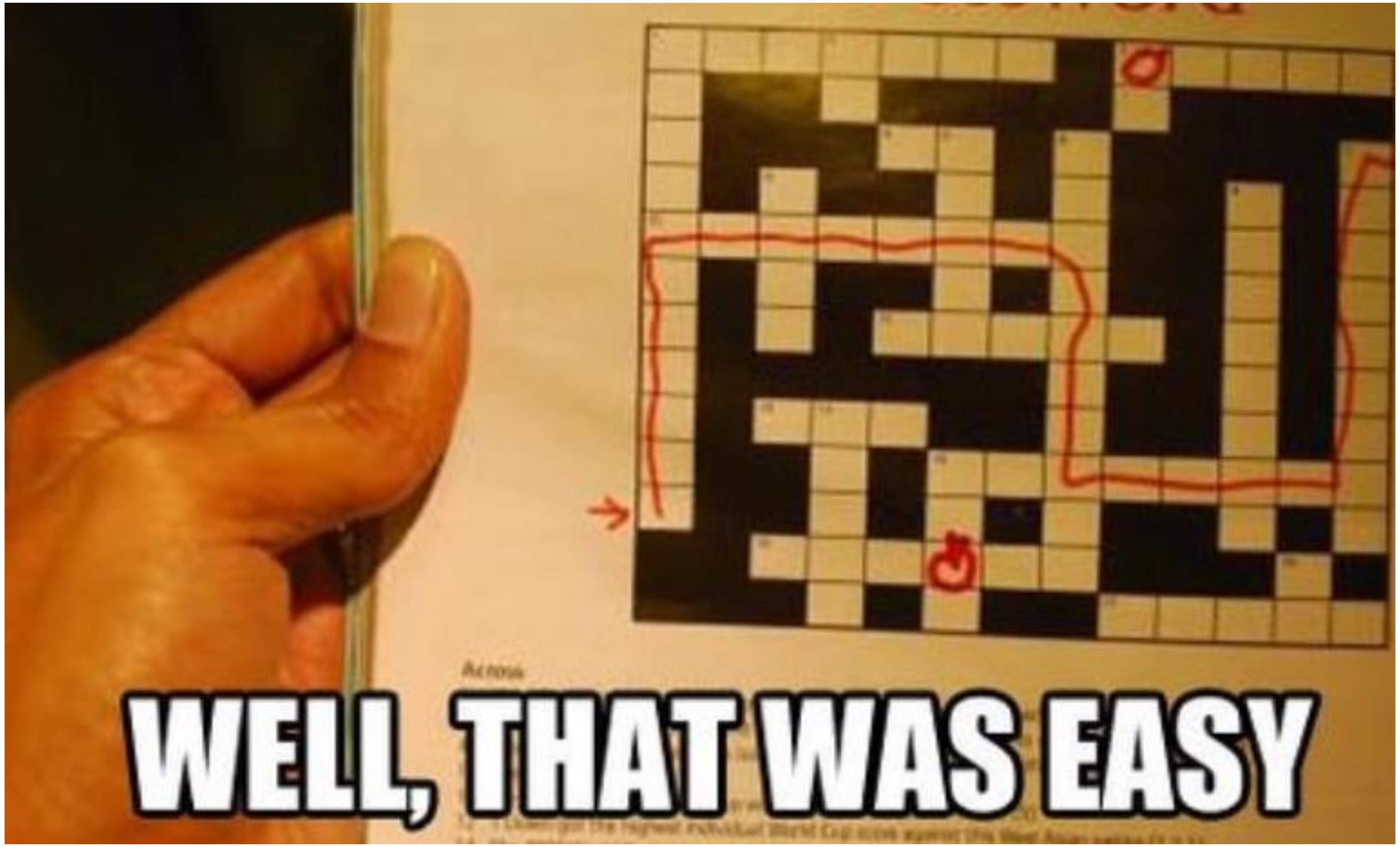
```
@FragmentationLevel1 = 5,
@FragmentationLevel2 = 30
```

### B. Rebuild or reorganize all indexes with fragmentation and update modified statistics on all user databases

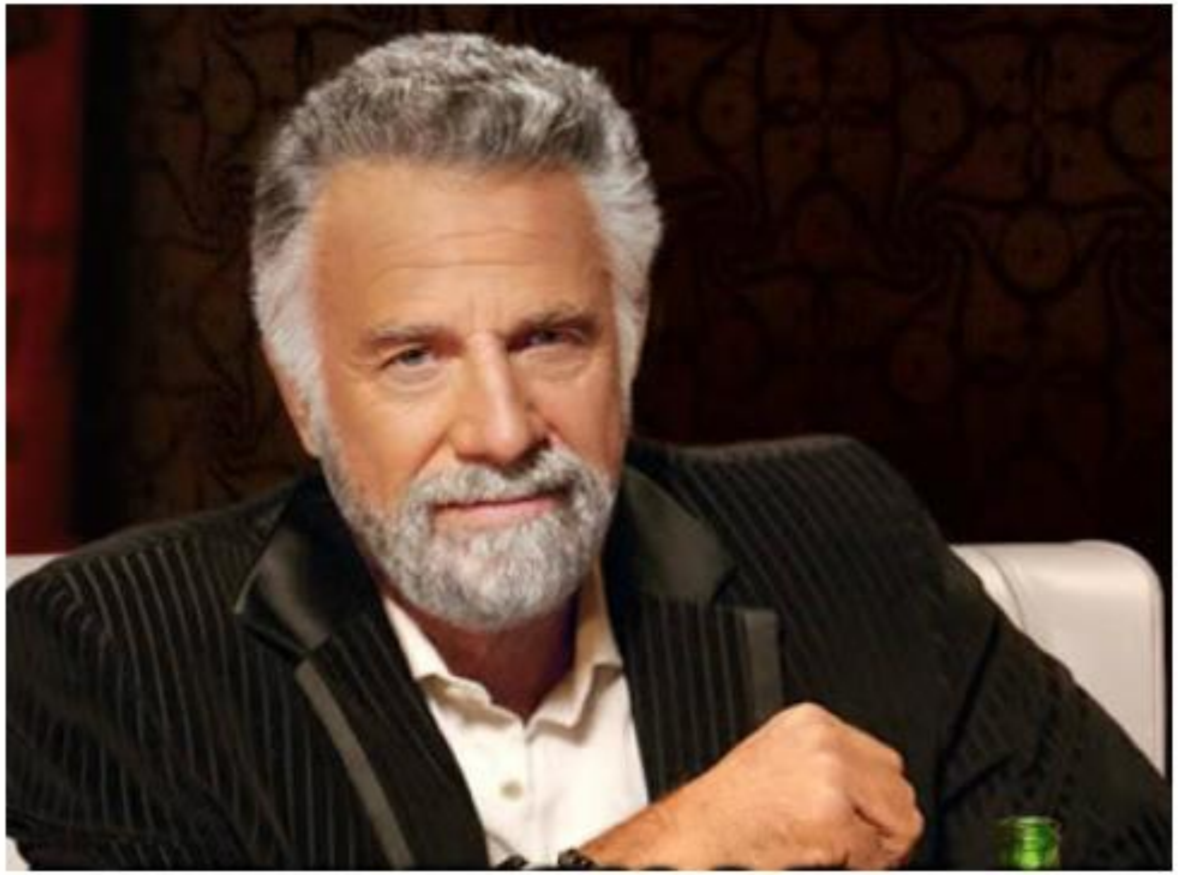
```
EXECUTE dbo.IndexOptimize
@Databases = 'USER_DATABASES',
@FragmentationLow = NULL,
@FragmentationMedium = 'INDEX_REORGANIZE,INDEX_REBUILD_ONLINE,INDEX_REBUILD_OFFLINE',
@FragmentationHigh = 'INDEX_REBUILD_ONLINE,INDEX_REBUILD_OFFLINE',
@FragmentationLevel1 = 5,
@FragmentationLevel2 = 30,
@UpdateStatistics = 'ALL',
@OnlyModifiedStatistics = 'Y'
```

### C. Update statistics on all user databases

# Indexes & Stats !! With OLA ..



I generally don't tune production queries !!!



But when I do , I do it better than developers !!!!

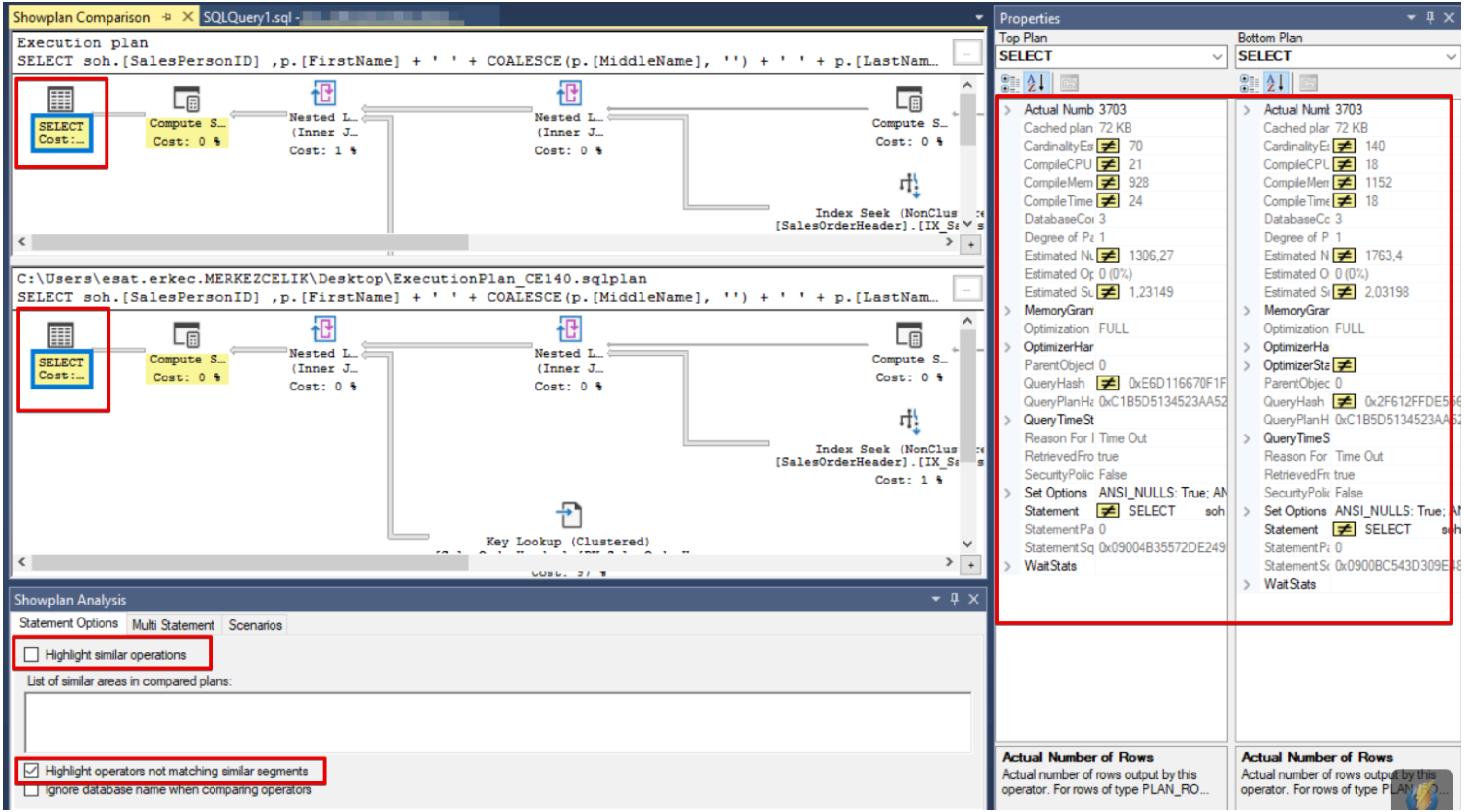
# Query Tuning best practices

- Use features \ tools like Live Query Stats, Query Store, Sentry Plan explorer, SSMS plan comparison etc.
- Use MAXDOP in hint if you want to force parallelism for better performance
- Keep primary key of lesser chars or integer. It is easier to process small width keys
- Use proper database types for the fields. If StartDate is database filed use date time as datatypes instead of VARCHAR (20).
- Use SP instead of Dynamic or Adhoc queries . The CPU time spent on calculating execution plan that can be eliminated, if we re-use the plan.
- **Avoid Using Scalar Functions in WHERE Clause on left side . Use on the right side if needed !!!**

e.g. SELECT Column FROM [dbo].[Table] WHERE dateadd(d,30,DateColumn) > getdate() ---- Not good

SELECT Column FROM [dbo].[Table] WHERE DateColumn > dateadd(d,-30,getdate()) ---- Good

- **Avoid Implicit conversions** between fields of different data types while comparing.
- **Avoid ORDER BY**; sort in the application layer instead. Consume all of the query results as fast as possible into memory in your app, and then sort.
- Yes, SCANS are not good and SEEKS are good, but NOT ALWAYS !!! Need to analyze the context.



```
Execution plan
SELECT soh.[SalesPersonID] ,p.[FirstName] + ' ' + COALESCE(p.[MiddleName], '') + ' ' + p.[LastNam…
```

```
C:\Users\esat.erkec.MERKEZCELIK\Desktop\ExecutionPlan_CE140.sqlplan
SELECT soh.[SalesPersonID] ,p.[FirstName] + ' ' + COALESCE(p.[MiddleName], '') + ' ' + p.[LastNam…
```

Top Plan: SELECT

| | |
|---|---|
| Actual Numb | 3703 |
| Cached plan | 72 KB |
| CardinalityEs | ≠ 70 |
| CompileCPU | ≠ 21 |
| CompileMem | ≠ 928 |
| CompileTime | ≠ 24 |
| DatabaseCo | 3 |
| Degree of Pa | 1 |
| Estimated Nu | ≠ 1306,27 |
| Estimated Op | 0 (0%) |
| Estimated Su | ≠ 1,23149 |
| MemoryGran | |
| Optimization | FULL |
| OptimizerHar | |
| ParentObject | 0 |
| QueryHash | ≠ 0xE6D116670F1F |
| QueryPlanHa | 0xC1B5D5134523AA52 |
| QueryTimeSt | |
| Reason For I | Time Out |
| RetrievedFro | true |
| SecurityPolic | False |
| Set Options | ANSI_NULLS: True; AN |
| Statement | ≠ SELECT soh |
| StatementPa | 0 |
| StatementSq | 0x09004B35572DE249 |
| WaitStats | |

Bottom Plan: SELECT

| | |
|---|---|
| Actual Numb | 3703 |
| Cached plar | 72 KB |
| CardinalityEs | ≠ 140 |
| CompileCPU | ≠ 18 |
| CompileMen | ≠ 1152 |
| CompileTime | ≠ 18 |
| DatabaseCc | 3 |
| Degree of P | 1 |
| Estimated N | ≠ 1763,4 |
| Estimated O | 0 (0%) |
| Estimated Si | ≠ 2,03198 |
| MemoryGrar | |
| Optimization | FULL |
| OptimizerHa | |
| OptimizerSta | ≠ |
| ParentObjec | 0 |
| QueryHash | ≠ 0x2F612FFDE56 |
| QueryPlanH | 0xC1B5D5134523AA5 |
| QueryTimeS | |
| Reason For | Time Out |
| RetrievedFr | true |
| SecurityPolic | False |
| Set Options | ANSI_NULLS: True; A |
| Statement | ≠ SELECT soh |
| StatementPa | 0 |
| StatementSc | 0x0900BC543D309E4 |
| WaitStats | |

**Actual Number of Rows**
Actual number of rows output by this operator. For rows of type PLAN_RO…

**Actual Number of Rows**
Actual number of rows output by this operator. For rows of type PLAN_RO…

Showplan Analysis

☐ Highlight similar operations

List of similar areas in compared plans:

☑ Highlight operators not matching similar segments

☐ Ignore database name when comparing operators

- Avoid The Bookmark Lookup operator uses a bookmark to look up the corresponding row in the table or clustered index.

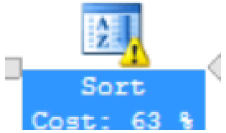
- The Key Lookup operator is a bookmark lookup on a table with a clustered index.

- RID Lookup is a bookmark lookup on a heap using a supplied row identifier (RID).

- The Spool operator saves an intermediate query result to the tempdb database.

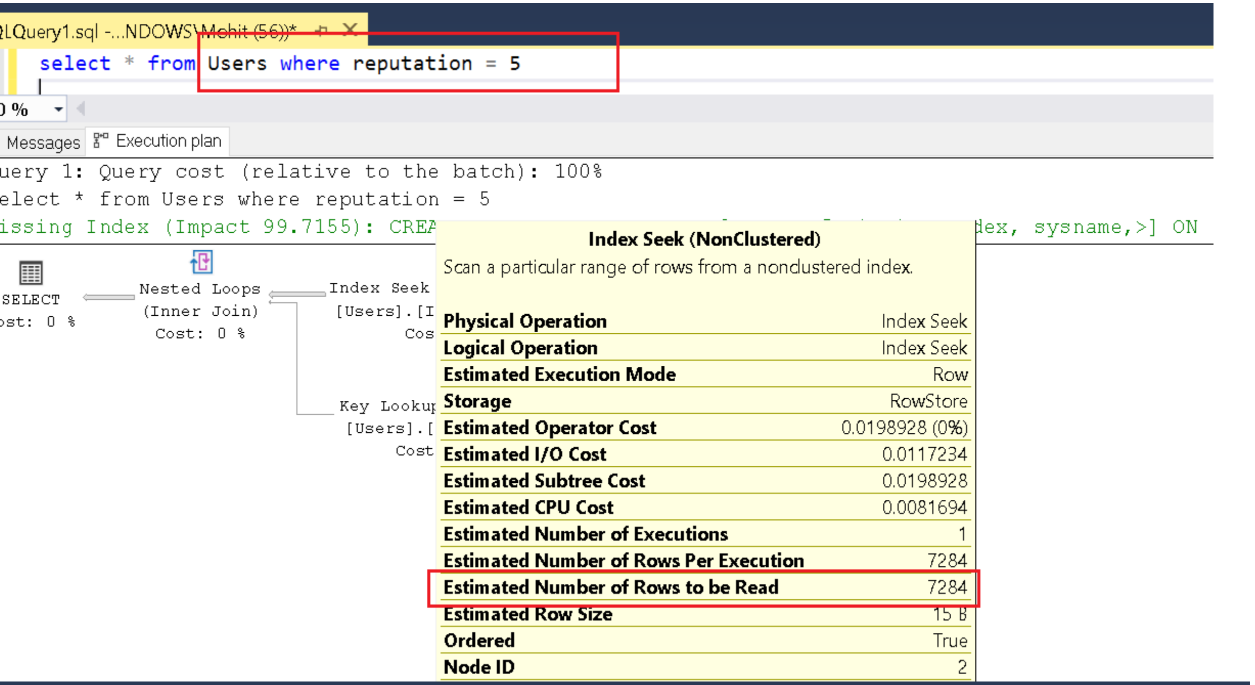
  Too much spools puts pressure on tempdb.

- **Tempdb\Memory spills while SORTs. Need to tune Stats**



- **If you see yellow bang sign on an operator.** , needs attention as it might be an impacting spill to disk basically.

DBCC SHOW_STATISITCS () tells as how the estimated number of rows are calculated for a plan. If this is not accurate, so will be the plan.



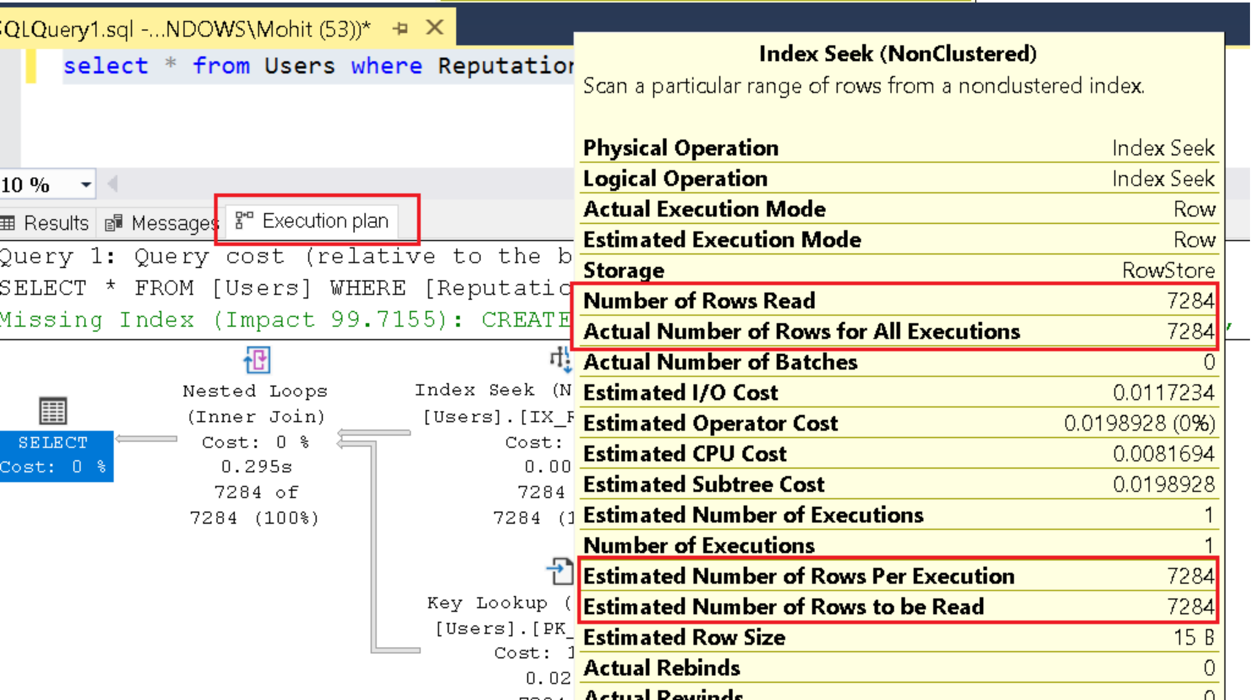
```
select * from Users where reputation = 5
```

```
uery 1: Query cost (relative to the batch): 100%
elect * from Users where reputation = 5
issing Index (Impact 99.7155): CREA                                  dex, sysname,>] ON
```

**Index Seek (NonClustered)**

Scan a particular range of rows from a nonclustered index.

| | |
|---|---|
| Physical Operation | Index Seek |
| Logical Operation | Index Seek |
| Estimated Execution Mode | Row |
| Storage | RowStore |
| Estimated Operator Cost | 0.0198928 (0%) |
| Estimated I/O Cost | 0.0117234 |
| Estimated Subtree Cost | 0.0198928 |
| Estimated CPU Cost | 0.0081694 |
| Estimated Number of Executions | 1 |
| Estimated Number of Rows Per Execution | 7284 |
| Estimated Number of Rows to be Read | 7284 |
| Estimated Row Size | 15 B |
| Ordered | True |
| Node ID | 2 |



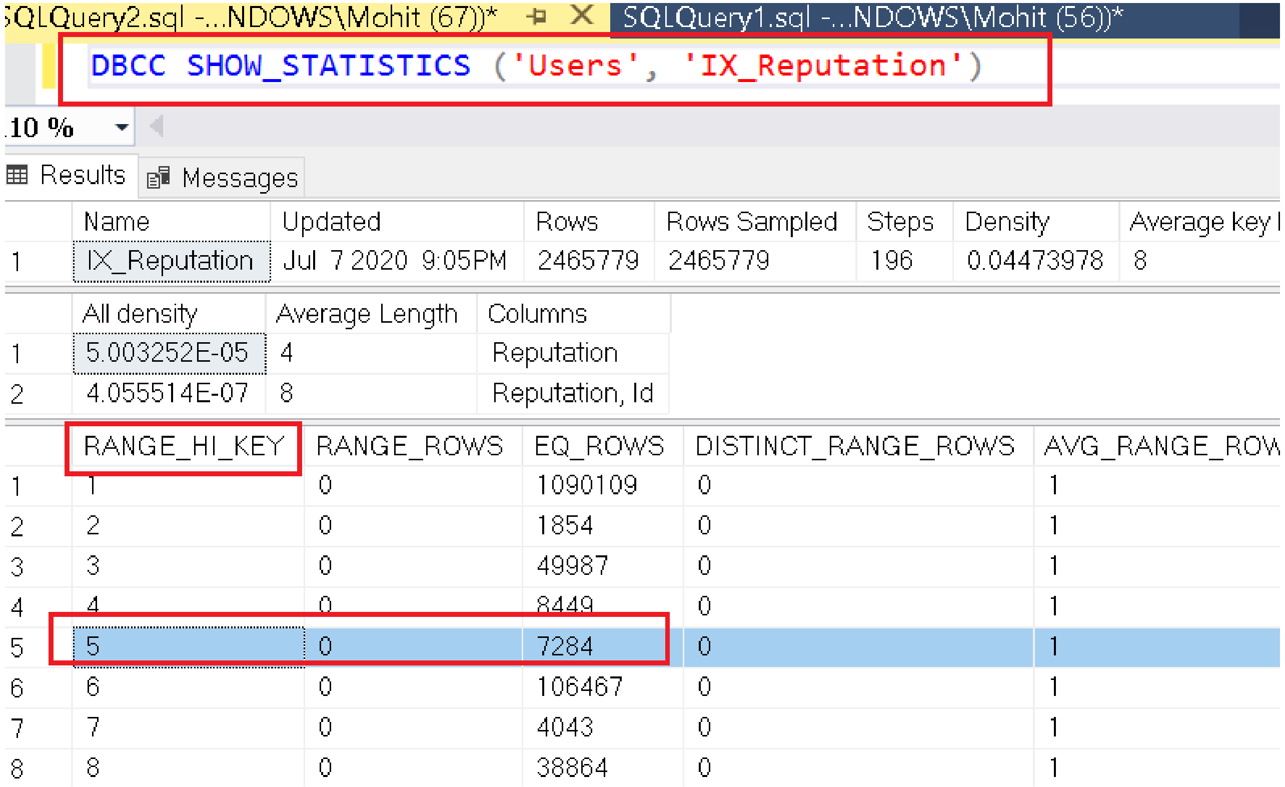
```
select * from Users where Reputation
```

```
Query 1: Query cost (relative to the b
SELECT * FROM [Users] WHERE [Reputatic
Missing Index (Impact 99.7155): CREATE
```

**Index Seek (NonClustered)**

Scan a particular range of rows from a nonclustered index.

| | |
|---|---|
| Physical Operation | Index Seek |
| Logical Operation | Index Seek |
| Actual Execution Mode | Row |
| Estimated Execution Mode | Row |
| Storage | RowStore |
| Number of Rows Read | 7284 |
| Actual Number of Rows for All Executions | 7284 |
| Actual Number of Batches | 0 |
| Estimated I/O Cost | 0.0117234 |
| Estimated Operator Cost | 0.0198928 (0%) |
| Estimated CPU Cost | 0.0081694 |
| Estimated Subtree Cost | 0.0198928 |
| Estimated Number of Executions | 1 |
| Number of Executions | 1 |
| Estimated Number of Rows Per Execution | 7284 |
| Estimated Number of Rows to be Read | 7284 |
| Estimated Row Size | 15 B |
| Actual Rebinds | 0 |
| Actual Rewinds | 0 |


SQLQuery2.sql -...NDOWS\Mohit (67))*
SQLQuery1.sql -...NDOWS\Mohit (56))*
10 %
Results
Messages

```
DBCC SHOW_STATISTICS ('Users', 'IX_Reputation')
```

| | Name | Updated | Rows | Rows Sampled | Steps | Density | Average key l |
|---|---|---|---|---|---|---|---|
| 1 | IX_Reputation | Jul 7 2020 9:05PM | 2465779 | 2465779 | 196 | 0.04473978 | 8 |

| | All density | Average Length | Columns |
|---|---|---|---|
| 1 | 5.003252E-05 | 4 | Reputation |
| 2 | 4.055514E-07 | 8 | Reputation, Id |

| | RANGE_HI_KEY | RANGE_ROWS | EQ_ROWS | DISTINCT_RANGE_ROWS | AVG_RANGE_ROW |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 1090109 | 0 | 1 |
| 2 | 2 | 0 | 1854 | 0 | 1 |
| 3 | 3 | 0 | 49987 | 0 | 1 |
| 4 | 4 | 0 | 8449 | 0 | 1 |
| 5 | 5 | 0 | 7284 | 0 | 1 |
| 6 | 6 | 0 | 106467 | 0 | 1 |
| 7 | 7 | 0 | 4043 | 0 | 1 |
| 8 | 8 | 0 | 38864 | 0 | 1 |

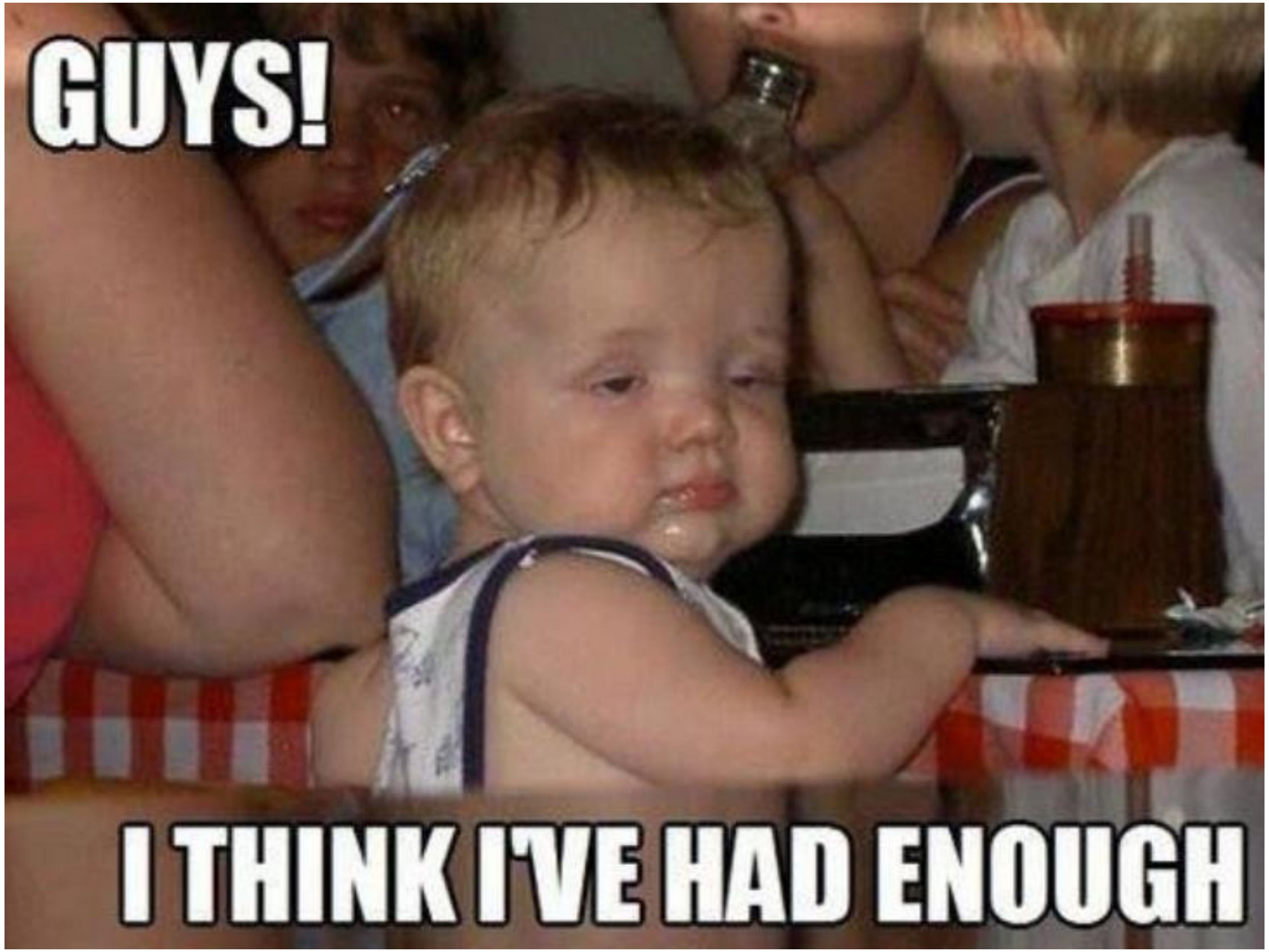
- **Avoid OR** in the Join Predicate/WHERE Clause Across Multiple Columns, since each component of the OR must be evaluated independently. Overhead !
- Avoid string searched using LIKE – WHERE Person.LastName LIKE **'%For%'**; Use LIKE **'For%'**;
- Use full text search if required instead of SARG .
- LARGE WRITES – Reduce the rows modified per operation would save locking , log usage.

**USE BATCH for heavy UPDATE or DELETE**

- For very heavily used queries, consider an indexed view to streamline constant access to important data.
- SELECT fields names , instead of using SELECT *
- Avoid CROSS JOINS or Cartesian products , big no UNLESS JUSTIFIED. Instead try to use INNER or OUTER JOINS.
- Break a large query into smaller queries whose data sets can later be joined together when ready.
- Missing indexes by the Query execution engine in plans but be careful before directly creating them.
  - Are there any existing indexes that are similar to this one that could be modified to cover this use case?
  - Do we need all of the include columns?
  - How high is the impact of the index? Will it improve a query by 98%, or only 5%.

- Try for SET STATISTICS IO ON, TIME ON - during tuning to get less Logical or Physical reads , CPU time .
- **CURSOR - BIG NO** , unless properly justified for existence.
- **Avoid Table variables .** Use CTE or temp table. Since before SQL 2019 @table is not taken properly in estimation. Only 1 row.
- **This SQL optimization prefers use of EXISTS().** If you want to check if a record exists, use EXISTS() instead of COUNT(). While COUNT() scans the entire table, EXISTS is light weight.
- Avoid nesting functions if possible. This can be confusing and lead to challenging performance problems.
- Avoid triggers that call stored procedures or that perform too much business logic.
- Nested triggers are equally dangerous. They may lead to unstable situations or infinite loops.
- **SQL scalar UDFs rarely perform well, especially on large dataset.** Reuse code by putting it into functions, a great practice in the app tier, but huge performance drawbacks in the database tier.
- **Use WHERE instead of HAVING to define filters.** HAVING statements are calculated after WHERE statements. If the intent is filter a query based on conditions, WHERE statement is more efficient.
- Use CTE instead of #TEMP or Table variable wherever possible, specially for comparatively small result set.

# Enough of theory !!! Feeling Sleepy ..Drowsy ??



# Types of Performance Issues

- Connectivity related issues
- Immediate blocking related issues
- Server\Instance resource usage related issues
- Database objects related issues - Indexes, Stats
- Code related issues - Bad query,  Parameter Sniffing, Parallelism

# Quick identification of the issue type

- It’s easy said than done ….. if we can find the root cause immediately then resolving that won’t be a problem…right !
- In many cases , identification itself takes time….

# Connectivity related issues

- Pretty easy !!!
- Check connectivity yourself, ask app team to confirm the connectivity , ping, nslookup, telnet, tracert etc…
- Can involve Network team to help further .

# Immediate blocking related issues

- Easy to identify !!!
- Check with related app team to terminate the culprit SPID and you are good to go….

# Server\Instance resource usage related issues

- Check the CPU , Memory etc. using Task Manager, Activity Monitor, Custom scripts
- May be a maintenance job running during the business hours, backup overshot run time etc.
- Wait stats related issues
- Sometimes not easy to find and resolve !!!!

# When the issue is not immediately identifiable ?

- Database objects related issues - Indexes, Stats
- Code related issues - Bad query,  Parameter Sniffing, Unbalanced Parallelism
- **Imagine you are on a P1 critical call !!!**

And you are like !!!!     Where to start ??

Too many scripts, content available on web, in my collection !!!

**But which one to use effectively ??**

# Let me share my approach , experience & resources..

Few very effective scripts\resources like –

sp_WhoisActive, First Responder Kit, Glen Berry Diagnostics ,Query Store , DBATools etc.

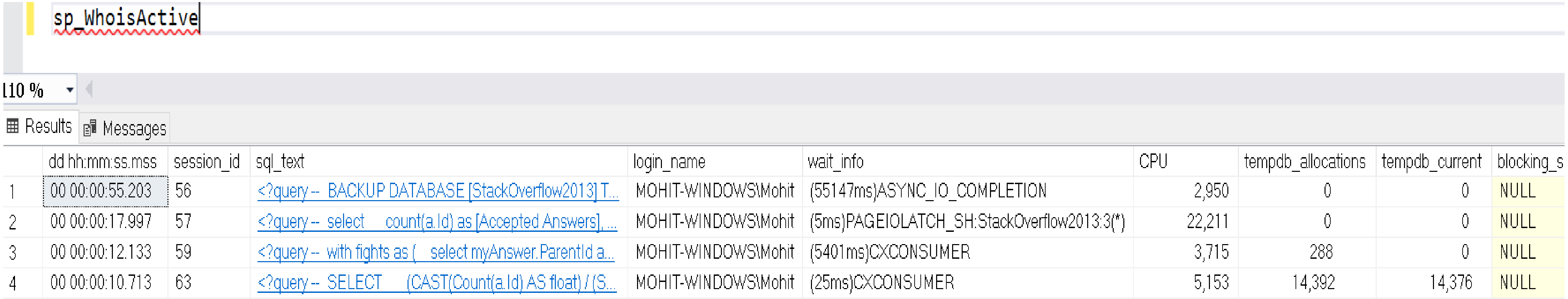
- **https://github.com/amachanic/sp_whoisactive/releases**
- **https://github.com/BrentOzarULTD/SQL-Server-First-Responder-Kit/tree/main**

- https://www.sqlskills.com/blogs/glenn/category/dmv-queries/
- https://github.com/ktaranov/sqlserver-kit
- https://github.com/JocaPC/qpi
- https://dbatools.io/

# Overview of sp_WhoisActive

[https://github.com/amachanic/sp_whoisactive/releases](https://github.com/amachanic/sp_whoisactive/releases)

- Open source , developed by Adam Mechanic
- Gives exhaustive information as compared to sp_who2  or other execution related DMVs.
- Very lightweight to run. No overhead.



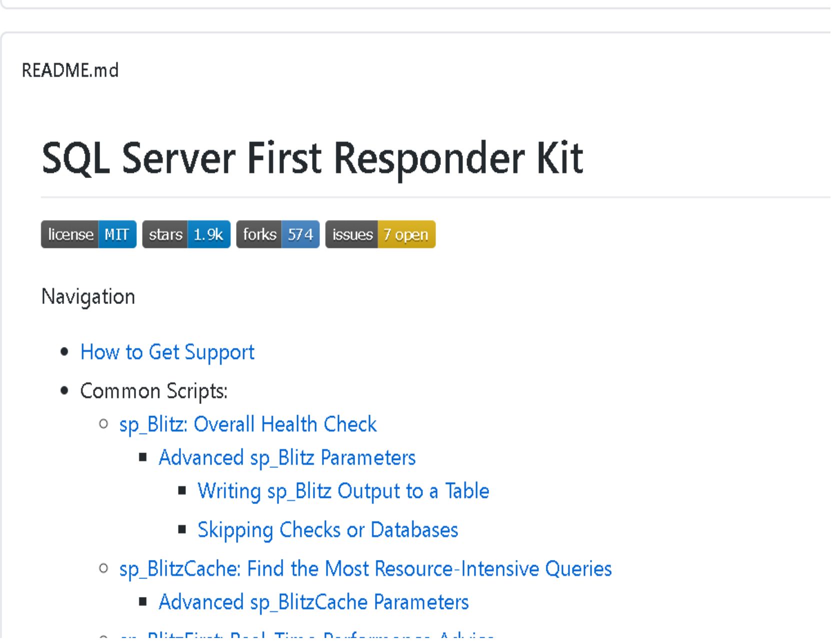
```
sp_WhoisActive
```

| | dd hh:mm:ss.mss | session_id | sql_text | login_name | wait_info | CPU | tempdb_allocations | tempdb_current | blocking_s |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 00 00:00:55.203 | 56 | <?query -- BACKUP DATABASE [StackOverflow2013] T... | MOHIT-WINDOWS\Mohit | (55147ms)ASYNC_IO_COMPLETION | 2,950 | 0 | 0 | NULL |
| 2 | 00 00:00:17.997 | 57 | <?query -- select    count(a.Id) as [Accepted Answers], ... | MOHIT-WINDOWS\Mohit | (5ms)PAGEIOLATCH_SH:StackOverflow2013:3(*) | 22,211 | 0 | 0 | NULL |
| 3 | 00 00:00:12.133 | 59 | <?query -- with fights as (   select myAnswer.ParentId a... | MOHIT-WINDOWS\Mohit | (5401ms)CXCONSUMER | 3,715 | 288 | 0 | NULL |
| 4 | 00 00:00:10.713 | 63 | <?query -- SELECT    (CAST(Count(a.Id) AS float) / (S... | MOHIT-WINDOWS\Mohit | (25ms)CXCONSUMER | 5,153 | 14,392 | 14,376 | NULL |

# Overview of the First Responder Kit

https://github.com/BrentOzarULTD/SQL-Server-First-Responder-Kit/tree/main

- Developed by Brent Ozar.
- Open source & Light weight.
- Supported on SQL 2012 onwards
- SQL Server 2008, 200R2 - not officially supported
- AWS RDS SQL Server - fully supported.
- Can be used to record data for historical analysis too by scheduling under jobs
- **Deploy using #sp_BlitzFirst as temporary procs if not allowed to create on server\master.**
- Some useful commands shown further ….



README.md

## SQL Server First Responder Kit

license MIT | stars 1.9k | forks 574 | issues 7 open

Navigation

- How to Get Support
- Common Scripts:
  - sp_Blitz: Overall Health Check
    - Advanced sp_Blitz Parameters
      - Writing sp_Blitz Output to a Table
      - Skipping Checks or Databases
  - sp_BlitzCache: Find the Most Resource-Intensive Queries
    - Advanced sp_BlitzCache Parameters
  - sp_BlitzFirst: Real-Time Performance Advice

# sp_BlitzFirst : Real-Time Performance Advice

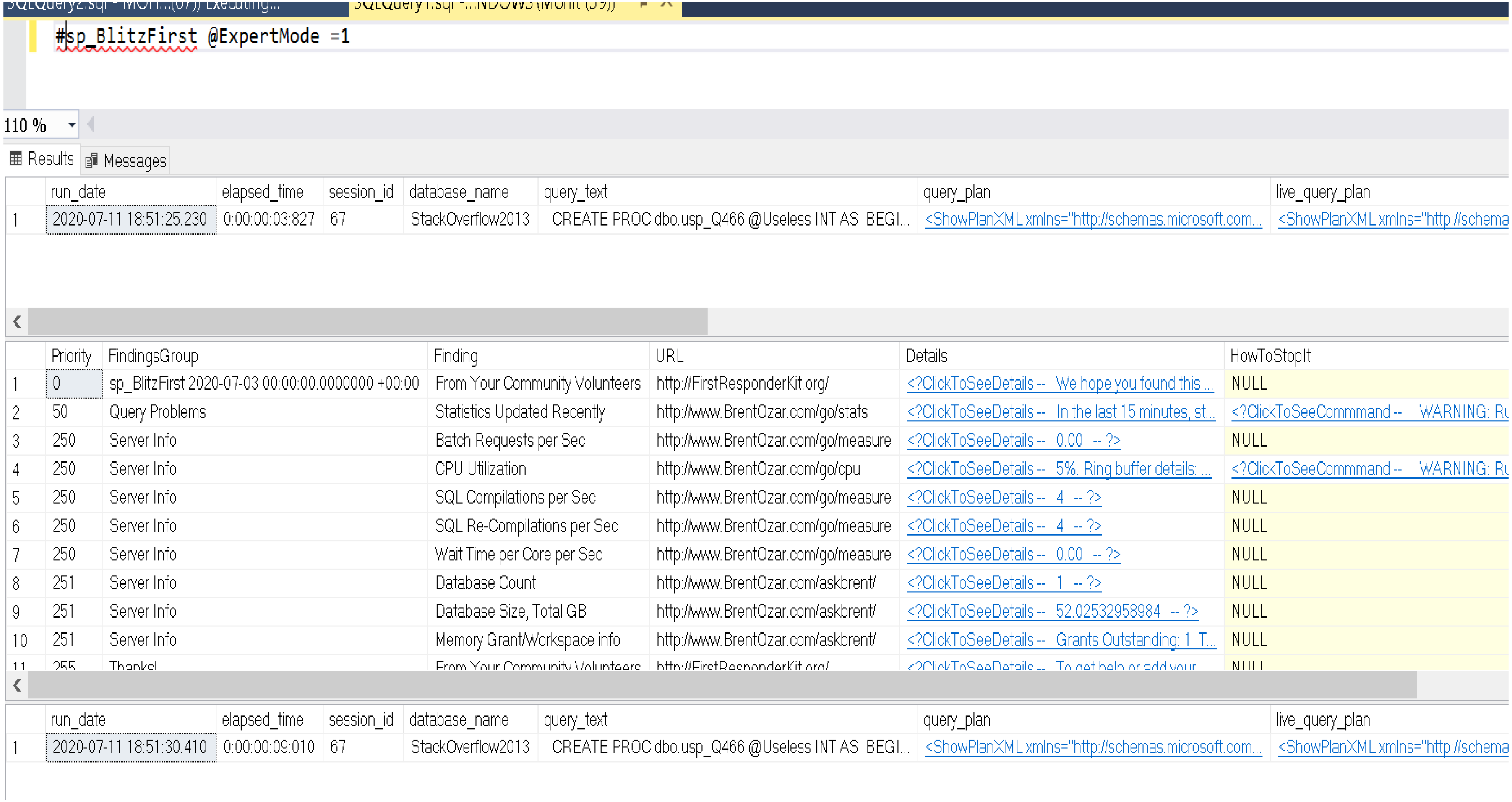
- It takes a sample from a bunch of DMVs , waits 5 seconds and then takes another sample.
- It examines the differences between the samples, and then gives you a prioritized list of things that might be causing performance issues right now.

Commonly used parameters:

- **@ExpertMode = 0 by default. If set to 1, will give more details.**
- You can dump data to table if required.

sp_BlitzFirst @OutputDatabaseName = 'DBAtools', @OutputSchemaName = 'dbo', @OutputTableName =   'BlitzFirstResults’;

--- Can be used for data capture for analysis.


SQLQuery2.sql - MOH...(67)) Executing...
SQLQuery1.sql -...NDOWS\Moh (59))
110 %
Results
Messages

```
#sp_BlitzFirst @ExpertMode =1
```

| | run_date | elapsed_time | session_id | database_name | query_text | query_plan | live_query_plan |
|---|---|---|---|---|---|---|---|
| 1 | 2020-07-11 18:51:25.230 | 0:00:00:03:827 | 67 | StackOverflow2013 | CREATE PROC dbo.usp_Q466 @Useless INT AS BEGI... | <ShowPlanXML xmlns="http://schemas.microsoft.com... | <ShowPlanXML xmlns="http://schema |

| | Priority | FindingsGroup | Finding | URL | Details | HowToStopIt |
|---|---|---|---|---|---|---|
| 1 | 0 | sp_BlitzFirst 2020-07-03 00:00:00.0000000 +00:00 | From Your Community Volunteers | http://FirstResponderKit.org/ | <?ClickToSeeDetails -- We hope you found this ... | NULL |
| 2 | 50 | Query Problems | Statistics Updated Recently | http://www.BrentOzar.com/go/stats | <?ClickToSeeDetails -- In the last 15 minutes, st... | <?ClickToSeeCommmand -- WARNING: Ru |
| 3 | 250 | Server Info | Batch Requests per Sec | http://www.BrentOzar.com/go/measure | <?ClickToSeeDetails -- 0.00 -- ?> | NULL |
| 4 | 250 | Server Info | CPU Utilization | http://www.BrentOzar.com/go/cpu | <?ClickToSeeDetails -- 5%. Ring buffer details: ... | <?ClickToSeeCommmand -- WARNING: Ru |
| 5 | 250 | Server Info | SQL Compilations per Sec | http://www.BrentOzar.com/go/measure | <?ClickToSeeDetails -- 4 -- ?> | NULL |
| 6 | 250 | Server Info | SQL Re-Compilations per Sec | http://www.BrentOzar.com/go/measure | <?ClickToSeeDetails -- 4 -- ?> | NULL |
| 7 | 250 | Server Info | Wait Time per Core per Sec | http://www.BrentOzar.com/go/measure | <?ClickToSeeDetails -- 0.00 -- ?> | NULL |
| 8 | 251 | Server Info | Database Count | http://www.BrentOzar.com/askbrent/ | <?ClickToSeeDetails -- 1 -- ?> | NULL |
| 9 | 251 | Server Info | Database Size, Total GB | http://www.BrentOzar.com/askbrent/ | <?ClickToSeeDetails -- 52.02532958984 -- ?> | NULL |
| 10 | 251 | Server Info | Memory Grant/Workspace info | http://www.BrentOzar.com/askbrent/ | <?ClickToSeeDetails -- Grants Outstanding: 1 T... | NULL |
| 11 | 255 | Thanks! | From Your Community Volunteers | http://FirstResponderKit.org/ | <?ClickToSeeDetails -- To get help or add your | NULL |

| | run_date | elapsed_time | session_id | database_name | query_text | query_plan | live_query_plan |
|---|---|---|---|---|---|---|---|
| 1 | 2020-07-11 18:51:30.410 | 0:00:00:09:010 | 67 | StackOverflow2013 | CREATE PROC dbo.usp_Q466 @Useless INT AS BEGI... | <ShowPlanXML xmlns="http://schemas.microsoft.com... | <ShowPlanXML xmlns="http://schema |

# sp_Blitz : Overall Health Check

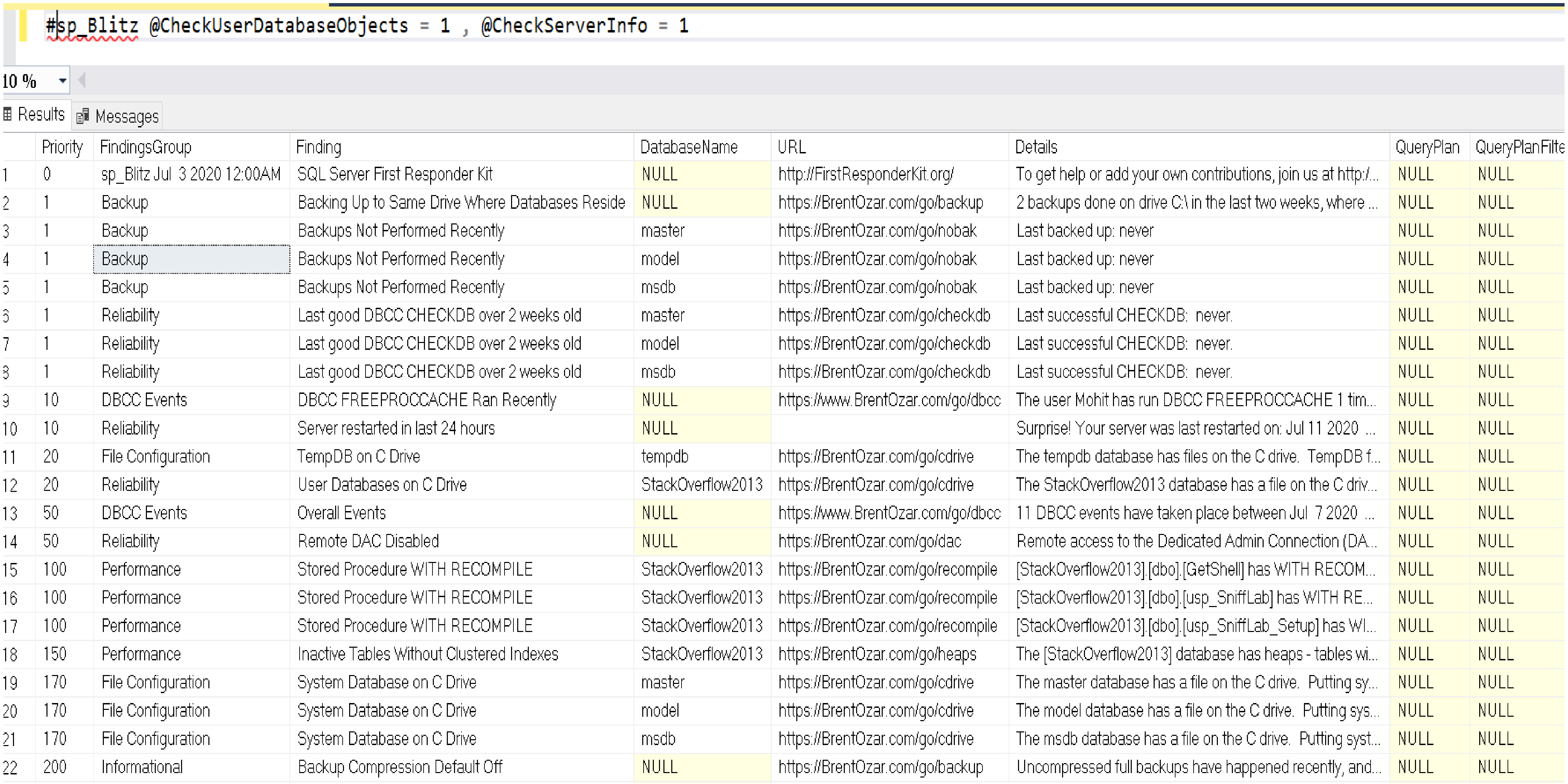
- Priority  1 - 50 is the most urgent, stuff that could get you fired. The warnings get progressively less urgent.

Commonly used parameters:

- @CheckUserDatabaseObjects = 0 - by default, it checks inside user databases for things like triggers or heaps. Turn this off (0) to ignore stuff you can't fix if you're managing third party databases.
- @CheckServerInfo = 1 - includes additional rows at priority 250 with server configuration details like service accounts.

- **Writing to a table directly :**

sp_Blitz @OutputDatabaseName = 'DBAtools', @OutputSchemaName = 'dbo', @OutputTableName = 'BlitzResults';



```
#sp_Blitz @CheckUserDatabaseObjects = 1 , @CheckServerInfo = 1
```

| | Priority | FindingsGroup | Finding | DatabaseName | URL | Details | QueryPlan | QueryPlanFilte |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | sp_Blitz Jul 3 2020 12:00AM | SQL Server First Responder Kit | NULL | http://FirstResponderKit.org/ | To get help or add your own contributions, join us at http:/... | NULL | NULL |
| 2 | 1 | Backup | Backing Up to Same Drive Where Databases Reside | NULL | https://BrentOzar.com/go/backup | 2 backups done on drive C:\ in the last two weeks, where ... | NULL | NULL |
| 3 | 1 | Backup | Backups Not Performed Recently | master | https://BrentOzar.com/go/nobak | Last backed up: never | NULL | NULL |
| 4 | 1 | Backup | Backups Not Performed Recently | model | https://BrentOzar.com/go/nobak | Last backed up: never | NULL | NULL |
| 5 | 1 | Backup | Backups Not Performed Recently | msdb | https://BrentOzar.com/go/nobak | Last backed up: never | NULL | NULL |
| 6 | 1 | Reliability | Last good DBCC CHECKDB over 2 weeks old | master | https://BrentOzar.com/go/checkdb | Last successful CHECKDB:  never. | NULL | NULL |
| 7 | 1 | Reliability | Last good DBCC CHECKDB over 2 weeks old | model | https://BrentOzar.com/go/checkdb | Last successful CHECKDB:  never. | NULL | NULL |
| 8 | 1 | Reliability | Last good DBCC CHECKDB over 2 weeks old | msdb | https://BrentOzar.com/go/checkdb | Last successful CHECKDB:  never. | NULL | NULL |
| 9 | 10 | DBCC Events | DBCC FREEPROCCACHE Ran Recently | NULL | https://www.BrentOzar.com/go/dbcc | The user Mohit has run DBCC FREEPROCCACHE 1 tim... | NULL | NULL |
| 10 | 10 | Reliability | Server restarted in last 24 hours | NULL | | Surprise! Your server was last restarted on: Jul 11 2020 ... | NULL | NULL |
| 11 | 20 | File Configuration | TempDB on C Drive | tempdb | https://BrentOzar.com/go/cdrive | The tempdb database has files on the C drive.  TempDB f... | NULL | NULL |
| 12 | 20 | Reliability | User Databases on C Drive | StackOverflow2013 | https://BrentOzar.com/go/cdrive | The StackOverflow2013 database has a file on the C driv... | NULL | NULL |
| 13 | 50 | DBCC Events | Overall Events | NULL | https://www.BrentOzar.com/go/dbcc | 11 DBCC events have taken place between Jul  7 2020 ... | NULL | NULL |
| 14 | 50 | Reliability | Remote DAC Disabled | NULL | https://BrentOzar.com/go/dac | Remote access to the Dedicated Admin Connection (DA... | NULL | NULL |
| 15 | 100 | Performance | Stored Procedure WITH RECOMPILE | StackOverflow2013 | https://BrentOzar.com/go/recompile | [StackOverflow2013].[dbo].[GetShell] has WITH RECOM... | NULL | NULL |
| 16 | 100 | Performance | Stored Procedure WITH RECOMPILE | StackOverflow2013 | https://BrentOzar.com/go/recompile | [StackOverflow2013].[dbo].[usp_SniffLab] has WITH RE... | NULL | NULL |
| 17 | 100 | Performance | Stored Procedure WITH RECOMPILE | StackOverflow2013 | https://BrentOzar.com/go/recompile | [StackOverflow2013].[dbo].[usp_SniffLab_Setup] has WI... | NULL | NULL |
| 18 | 150 | Performance | Inactive Tables Without Clustered Indexes | StackOverflow2013 | https://BrentOzar.com/go/heaps | The [StackOverflow2013] database has heaps - tables wi... | NULL | NULL |
| 19 | 170 | File Configuration | System Database on C Drive | master | https://BrentOzar.com/go/cdrive | The master database has a file on the C drive.  Putting sy... | NULL | NULL |
| 20 | 170 | File Configuration | System Database on C Drive | model | https://BrentOzar.com/go/cdrive | The model database has a file on the C drive.  Putting sys... | NULL | NULL |
| 21 | 170 | File Configuration | System Database on C Drive | msdb | https://BrentOzar.com/go/cdrive | The msdb database has a file on the C drive.  Putting syst... | NULL | NULL |
| 22 | 200 | Informational | Backup Compression Default Off | NULL | https://BrentOzar.com/go/backup | Uncompressed full backups have happened recently, and... | NULL | NULL |

# sp_BlitzCache : Find the Most Resource-Intensive Queries

sp_BlitzCache looks at your plan cache where SQL Server keeps track of which queries have run recently, and how much impact they've had on the server.

- The first result set shows your 10 most resource-intensive queries.
- The second result set explains the contents of the Warnings column - but it only shows the warnings that were produced in the first result set.

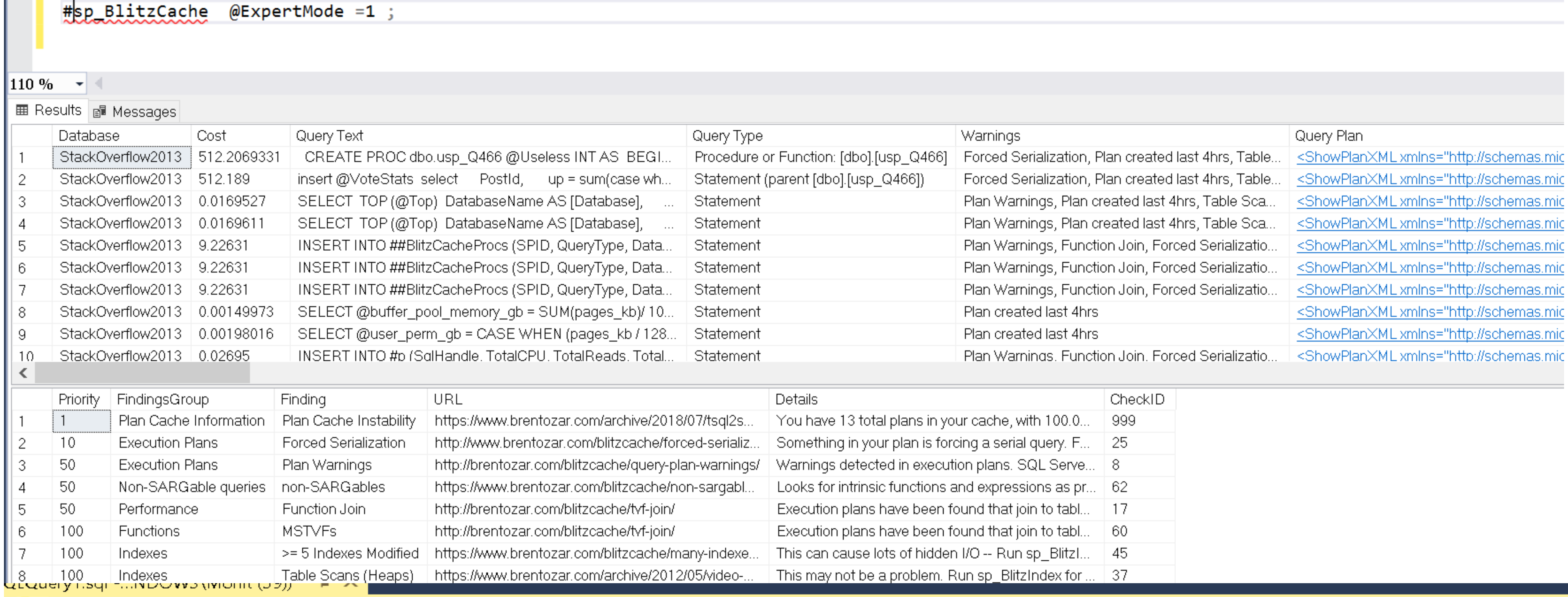
Common sp_BlitzCache Parameters

**@SortOrder** parameter lets you pick which top 10 queries you want to examine. Some of them :

- cpu
- reads
- writes

**@ExpertMode = 1 - turn this on**, and you get more columns with more data.

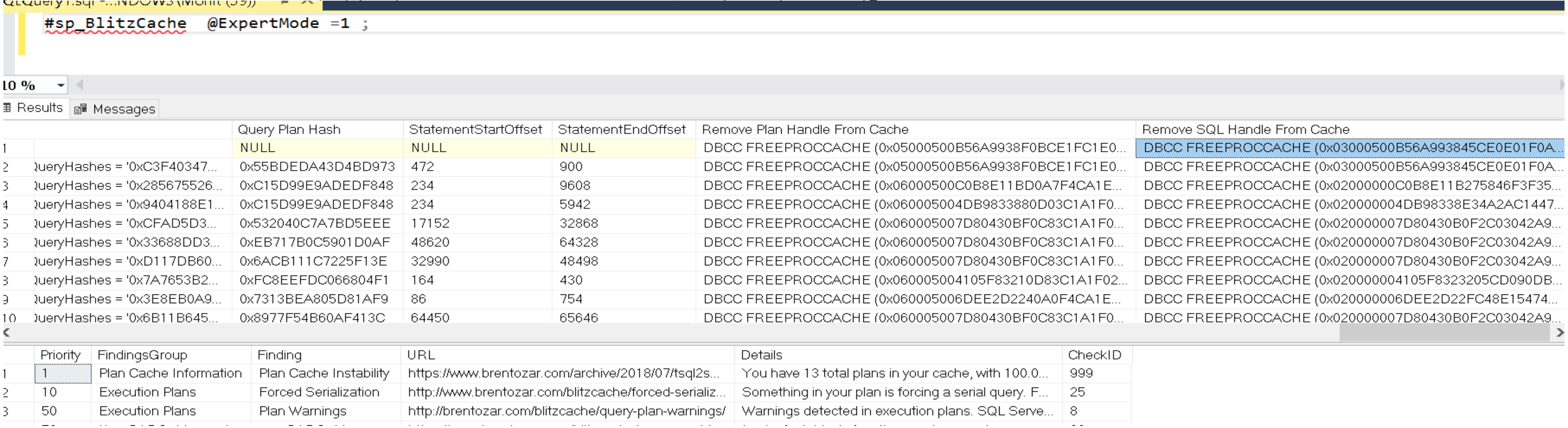
Example - sp_BlitzCache @ExpertMode = 1 , @SortOrder = ‘cpu’ ;



```
#sp_BlitzCache  @ExpertMode =1 ;
```

| | Database | Cost | Query Text | Query Type | Warnings | Query Plan |
|---|---|---|---|---|---|---|
| 1 | StackOverflow2013 | 512.2069331 | CREATE PROC dbo.usp_Q466 @Useless INT AS BEGI... | Procedure or Function: [dbo].[usp_Q466] | Forced Serialization, Plan created last 4hrs, Table... | <ShowPlanXML xmlns="http://schemas.mic |
| 2 | StackOverflow2013 | 512.189 | insert @VoteStats select PostId, up = sum(case wh... | Statement (parent [dbo].[usp_Q466]) | Forced Serialization, Plan created last 4hrs, Table... | <ShowPlanXML xmlns="http://schemas.mic |
| 3 | StackOverflow2013 | 0.0169527 | SELECT TOP (@Top) DatabaseName AS [Database], ... | Statement | Plan Warnings, Plan created last 4hrs, Table Sca... | <ShowPlanXML xmlns="http://schemas.mic |
| 4 | StackOverflow2013 | 0.0169611 | SELECT TOP (@Top) DatabaseName AS [Database], ... | Statement | Plan Warnings, Plan created last 4hrs, Table Sca... | <ShowPlanXML xmlns="http://schemas.mic |
| 5 | StackOverflow2013 | 9.22631 | INSERT INTO ##BlitzCacheProcs (SPID, QueryType, Data... | Statement | Plan Warnings, Function Join, Forced Serializatio... | <ShowPlanXML xmlns="http://schemas.mic |
| 6 | StackOverflow2013 | 9.22631 | INSERT INTO ##BlitzCacheProcs (SPID, QueryType, Data... | Statement | Plan Warnings, Function Join, Forced Serializatio... | <ShowPlanXML xmlns="http://schemas.mic |
| 7 | StackOverflow2013 | 9.22631 | INSERT INTO ##BlitzCacheProcs (SPID, QueryType, Data... | Statement | Plan Warnings, Function Join, Forced Serializatio... | <ShowPlanXML xmlns="http://schemas.mic |
| 8 | StackOverflow2013 | 0.00149973 | SELECT @buffer_pool_memory_gb = SUM(pages_kb)/ 10... | Statement | Plan created last 4hrs | <ShowPlanXML xmlns="http://schemas.mic |
| 9 | StackOverflow2013 | 0.00198016 | SELECT @user_perm_gb = CASE WHEN (pages_kb / 128... | Statement | Plan created last 4hrs | <ShowPlanXML xmlns="http://schemas.mic |
| 10 | StackOverflow2013 | 0.02695 | INSERT INTO #p (SqlHandle, TotalCPU, TotalReads, Total... | Statement | Plan Warnings, Function Join, Forced Serializatio... | <ShowPlanXML xmlns="http://schemas.mic |

| | Priority | FindingsGroup | Finding | URL | Details | CheckID |
|---|---|---|---|---|---|---|
| 1 | 1 | Plan Cache Information | Plan Cache Instability | https://www.brentozar.com/archive/2018/07/tsql2s... | You have 13 total plans in your cache, with 100.0... | 999 |
| 2 | 10 | Execution Plans | Forced Serialization | http://www.brentozar.com/blitzcache/forced-serializ... | Something in your plan is forcing a serial query. F... | 25 |
| 3 | 50 | Execution Plans | Plan Warnings | http://brentozar.com/blitzcache/query-plan-warnings/ | Warnings detected in execution plans. SQL Serve... | 8 |
| 4 | 50 | Non-SARGable queries | non-SARGables | https://www.brentozar.com/blitzcache/non-sargabl... | Looks for intrinsic functions and expressions as pr... | 62 |
| 5 | 50 | Performance | Function Join | http://brentozar.com/blitzcache/tvf-join/ | Execution plans have been found that join to tabl... | 17 |
| 6 | 100 | Functions | MSTVFs | http://brentozar.com/blitzcache/tvf-join/ | Execution plans have been found that join to tabl... | 60 |
| 7 | 100 | Indexes | >= 5 Indexes Modified | https://www.brentozar.com/blitzcache/many-indexe... | This can cause lots of hidden I/O -- Run sp_BlitzI... | 45 |
| 8 | 100 | Indexes | Table Scans (Heaps) | https://www.brentozar.com/archive/2012/05/video-... | This may not be a problem. Run sp_BlitzIndex for ... | 37 |



```
#sp_BlitzCache  @ExpertMode =1 ;
```

| | | Query Plan Hash | StatementStartOffset | StatementEndOffset | Remove Plan Handle From Cache | Remove SQL Handle From Cache |
|---|---|---|---|---|---|---|
| 1 | | NULL | NULL | NULL | DBCC FREEPROCCACHE (0x05000500B56A9938F0BCE1FC1E0... | DBCC FREEPROCCACHE (0x03000500B56A993845CE0E01F0A... |
| 2 | ueryHashes = '0xC3F40347... | 0x55BDEDA43D4BD973 | 472 | 900 | DBCC FREEPROCCACHE (0x05000500B56A9938F0BCE1FC1E0... | DBCC FREEPROCCACHE (0x03000500B56A993845CE0E01F0A... |
| 3 | ueryHashes = '0x285675526... | 0xC15D99E9ADEDF848 | 234 | 9608 | DBCC FREEPROCCACHE (0x06000500C0B8E11BD0A7F4CA1E... | DBCC FREEPROCCACHE (0x02000000C0B8E11B275846F3F35... |
| 4 | ueryHashes = '0x9404188E1... | 0xC15D99E9ADEDF848 | 234 | 5942 | DBCC FREEPROCCACHE (0x060005004DB9833880D03C1A1F0... | DBCC FREEPROCCACHE (0x020000004DB98338E34A2AC1447... |
| 5 | ueryHashes = '0xCFAD5D3... | 0x532040C7A7BD5EEE | 17152 | 32868 | DBCC FREEPROCCACHE (0x060005007D80430BF0C83C1A1F0... | DBCC FREEPROCCACHE (0x020000007D80430B0F2C03042A9... |
| 6 | ueryHashes = '0x33688DD3... | 0xEB717B0C5901D0AF | 48620 | 64328 | DBCC FREEPROCCACHE (0x060005007D80430BF0C83C1A1F0... | DBCC FREEPROCCACHE (0x020000007D80430B0F2C03042A9... |
| 7 | ueryHashes = '0xD117DB60... | 0x6ACB111C7225F13E | 32990 | 48498 | DBCC FREEPROCCACHE (0x060005007D80430BF0C83C1A1F0... | DBCC FREEPROCCACHE (0x020000007D80430B0F2C03042A9... |
| 8 | ueryHashes = '0x7A7653B2... | 0xFC8EEFDC066804F1 | 164 | 430 | DBCC FREEPROCCACHE (0x060005004105F83210D83C1A1F02... | DBCC FREEPROCCACHE (0x020000004105F8323205CD090DB... |
| 9 | ueryHashes = '0x3E8EB0A9... | 0x7313BEA805D81AF9 | 86 | 754 | DBCC FREEPROCCACHE (0x060005006DEE2D2240A0F4CA1E... | DBCC FREEPROCCACHE (0x020000006DEE2D22FC48E15474... |
| 10 | ueryHashes = '0x6B11B645... | 0x8977F54B60AF413C | 64450 | 65646 | DBCC FREEPROCCACHE (0x060005007D80430BF0C83C1A1F0... | DBCC FREEPROCCACHE (0x020000007D80430B0F2C03042A9... |

| | Priority | FindingsGroup | Finding | URL | Details | CheckID |
|---|---|---|---|---|---|---|
| 1 | 1 | Plan Cache Information | Plan Cache Instability | https://www.brentozar.com/archive/2018/07/tsql2s... | You have 13 total plans in your cache, with 100.0... | 999 |
| 2 | 10 | Execution Plans | Forced Serialization | http://www.brentozar.com/blitzcache/forced-serializ... | Something in your plan is forcing a serial query. F... | 25 |
| 3 | 50 | Execution Plans | Plan Warnings | http://brentozar.com/blitzcache/query-plan-warnings/ | Warnings detected in execution plans. SQL Serve... | 8 |

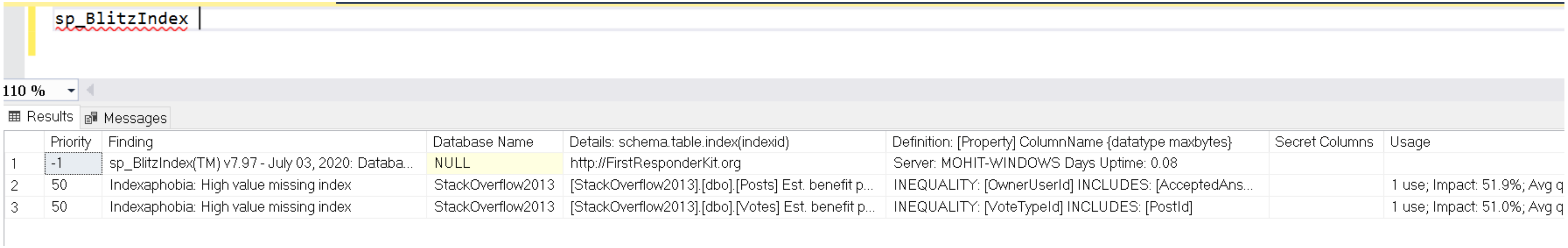
# sp_BlitzIndex : Tune Your Indexes

sp_BlitzIndex  analyzes the indexes of the database you're in (your current context.)

Common parameters include:

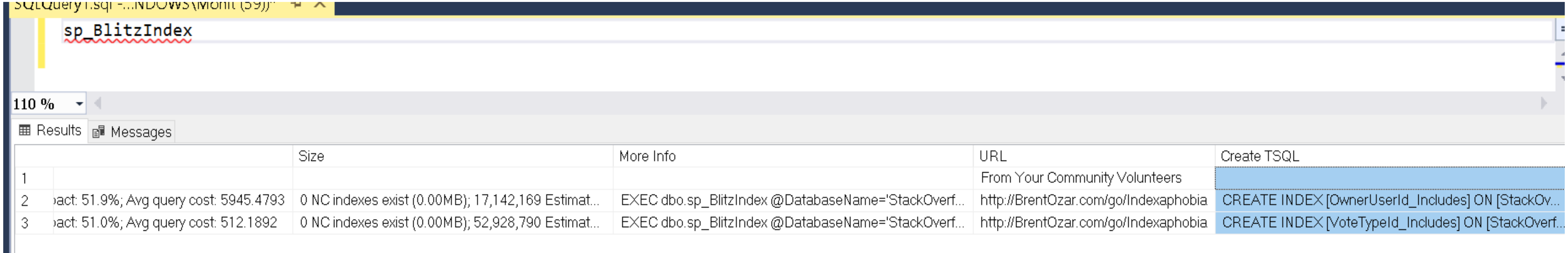
- @DatabaseName - if you want to analyze a specific database
- @SchemaName, @TableName - if you pass in these, sp_BlitzIndex does a deeper-dive analysis of just one table.
- @Mode = 0 (default)

Get different data with 0=Diagnose, 1=Summarize, 2=Index Usage Detail, 3=Missing Index Detail, 4=Diagnose Details.



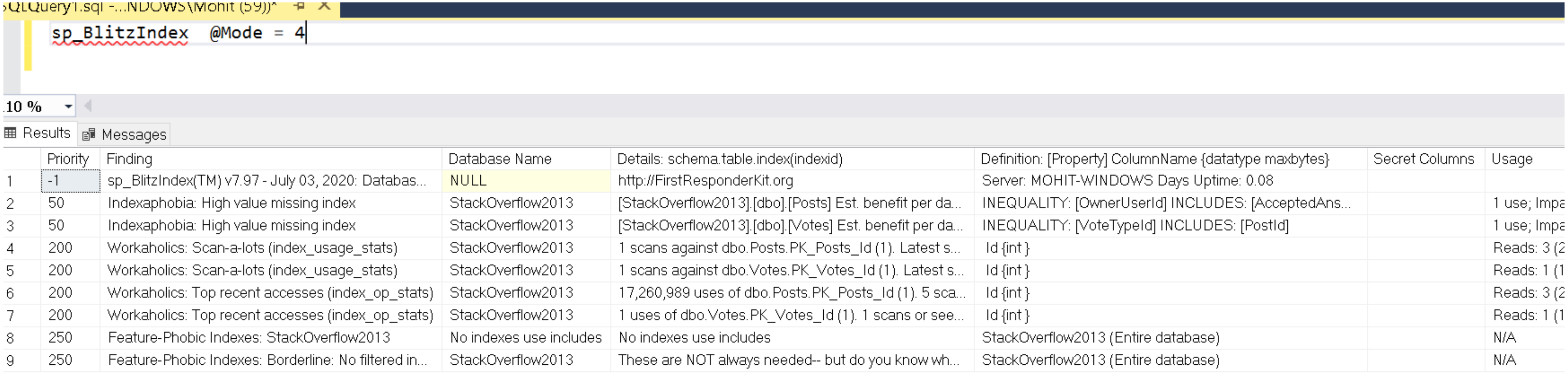
```
sp_BlitzIndex
```

| | Priority | Finding | Database Name | Details: schema.table.index(indexid) | Definition: [Property] ColumnName {datatype maxbytes} | Secret Columns | Usage |
|---|---|---|---|---|---|---|---|
| 1 | -1 | sp_BlitzIndex(TM) v7.97 - July 03, 2020: Databa... | NULL | http://FirstResponderKit.org | Server: MOHIT-WINDOWS Days Uptime: 0.08 | | |
| 2 | 50 | Indexaphobia: High value missing index | StackOverflow2013 | [StackOverflow2013].[dbo].[Posts] Est. benefit p... | INEQUALITY: [OwnerUserId] INCLUDES: [AcceptedAns... | | 1 use; Impact: 51.9%; Avg q |
| 3 | 50 | Indexaphobia: High value missing index | StackOverflow2013 | [StackOverflow2013].[dbo].[Votes] Est. benefit p... | INEQUALITY: [VoteTypeId] INCLUDES: [PostId] | | 1 use; Impact: 51.0%; Avg q |



```
sp_BlitzIndex
```

| | | Size | More Info | URL | Create TSQL |
|---|---|---|---|---|---|
| 1 | | | | From Your Community Volunteers | |
| 2 | pact: 51.9%; Avg query cost: 5945.4793 | 0 NC indexes exist (0.00MB); 17,142,169 Estimat... | EXEC dbo.sp_BlitzIndex @DatabaseName='StackOverf... | http://BrentOzar.com/go/Indexaphobia | CREATE INDEX [OwnerUserId_Includes] ON [StackOv... |
| 3 | pact: 51.0%; Avg query cost: 512.1892 | 0 NC indexes exist (0.00MB); 52,928,790 Estimat... | EXEC dbo.sp_BlitzIndex @DatabaseName='StackOverf... | http://BrentOzar.com/go/Indexaphobia | CREATE INDEX [VoteTypeId_Includes] ON [StackOverf... |



```
sp_BlitzIndex  @Mode = 4
```

| | Priority | Finding | Database Name | Details: schema.table.index(indexid) | Definition: [Property] ColumnName {datatype maxbytes} | Secret Columns | Usage |
|---|---|---|---|---|---|---|---|
| 1 | -1 | sp_BlitzIndex(TM) v7.97 - July 03, 2020: Databas... | NULL | http://FirstResponderKit.org | Server: MOHIT-WINDOWS Days Uptime: 0.08 | | |
| 2 | 50 | Indexaphobia: High value missing index | StackOverflow2013 | [StackOverflow2013].[dbo].[Posts] Est. benefit per da... | INEQUALITY: [OwnerUserId] INCLUDES: [AcceptedAns... | | 1 use; Impa |
| 3 | 50 | Indexaphobia: High value missing index | StackOverflow2013 | [StackOverflow2013].[dbo].[Votes] Est. benefit per da... | INEQUALITY: [VoteTypeId] INCLUDES: [PostId] | | 1 use; Impa |
| 4 | 200 | Workaholics: Scan-a-lots (index_usage_stats) | StackOverflow2013 | 1 scans against dbo.Posts.PK_Posts_Id (1). Latest s... | Id {int } | | Reads: 3 (2 |
| 5 | 200 | Workaholics: Scan-a-lots (index_usage_stats) | StackOverflow2013 | 1 scans against dbo.Votes.PK_Votes_Id (1). Latest s... | Id {int } | | Reads: 1 (1 |
| 6 | 200 | Workaholics: Top recent accesses (index_op_stats) | StackOverflow2013 | 17,260,989 uses of dbo.Posts.PK_Posts_Id (1). 5 sca... | Id {int } | | Reads: 3 (2 |
| 7 | 200 | Workaholics: Top recent accesses (index_op_stats) | StackOverflow2013 | 1 uses of dbo.Votes.PK_Votes_Id (1). 1 scans or see... | Id {int } | | Reads: 1 (1 |
| 8 | 250 | Feature-Phobic Indexes: StackOverflow2013 | No indexes use includes | No indexes use includes | StackOverflow2013 (Entire database) | | N/A |
| 9 | 250 | Feature-Phobic Indexes: Borderline: No filtered in... | StackOverflow2013 | These are NOT always needed-- but do you know wh... | StackOverflow2013 (Entire database) | | N/A |

# Other Blitz procedures

- sp_BlitzInMemoryOLTP :  In Memory OLTP Analysis.
- sp_BlitzLock : Deadlock Analysis.
- sp_BlitzQueryStore : How Has a Query Plan Changed Over Time.
- sp_BlitzBackups : How Much Data Could You Lose.

- For other procedures details , please refer the GitHub documentation.

- **Create the procedures as #sp_Blitz so that they are not created anywhere on the server , if not allowed.**

# Order of using the Blitz procedures during real time issue scenario !! Really helps….

1. Your monitoring software
2. sp_WhoisActive
3. sp_BlitzFirst
4. sp_Blitz
5. sp_BlitzCache
6. sp_BlitzIndex

# References

- https://github.com/BrentOzarULTD/SQL-Server-First-Responder-Kit/tree/main
- https://github.com/amachanic/sp_whoisactive/releases
- https://www.sqlskills.com/blogs/glenn/category/dmv-queries/
- https://github.com/ktaranov/sqlserver-kit
- https://github.com/JocaPC/qpi
- https://dbatools.io/
- https://www.brentozar.com/
- https://www.sqlskills.com/

CONGRATULATIONS
YOU DID IT !!

Thank you
Thank you very much