**GITHUB repository for SQL DBA scripts:**

[**https://github.com/praveenmadupu/DBAImpScripts/blob/main/TempDB%20usage%20Audit**](https://github.com/praveenmadupu/DBAImpScripts/blob/main/TempDB%20usage%20Audit)

[**https://github.com/praveenmadupu/SQLDBAInventory/blob/master/Performance%20-%20sp\_waitstatistics**](https://github.com/praveenmadupu/SQLDBAInventory/blob/master/Performance%20-%20sp_waitstatistics)

**https://github.com/rajendragp/RajendraScripts**

**Temp DB Usage Query:**

use [tempdb]

GO

select getdate() AS runtime, SUM(user\_object\_reserved\_page\_count)\*8 as usr\_obj\_kb,

SUM(internal\_object\_reserved\_page\_count)\*8 as internal\_obj\_kb,

SUM(version\_store\_reserved\_page\_count)\*8 as version\_store\_kb,

SUM(unallocated\_extent\_page\_count)\*8 as freespace\_kb,

SUM(mixed\_extent\_page\_count)\*8 as mixedextent\_kb

FROM sys.dm\_db\_file\_space\_usage

Navatha kattela

use [tempdb]

GO

Select name AS [Logical Name], size/128.0 AS [Total Size in MB],

size/128.0 - CAST(FILEPROPERTY(name, 'spaceused') AS int)/128.0 AS [Available space in MB] from sys.database\_files

**Query to find which transaction utilizing version store space:**

SELECT GETDATE() AS runtime

              ,a.\*

              ,b.kpid

              ,b.blocked

              ,b.lastwaittype

              ,b.waitresource

              ,db\_name(b.dbid) AS database\_name

              ,b.cpu

              ,b.physical\_io

              ,b.memusage

              ,b.login\_time

              ,b.last\_batch

              ,b.open\_tran

              ,b.STATUS

              ,b.hostname

              ,b.program\_name

              ,b.cmd

              ,b.loginame

              ,request\_id

FROM sys.dm\_tran\_active\_snapshot\_database\_transactions a

INNER JOIN sys.sysprocesses b ON a.session\_id = b.spid

--monitoring active sessions.

SELECT

s.session\_id,

s.login\_name,

s.host\_name,

s.status,

r.status AS request\_status,

r.command,

r.cpu\_time,

r.logical\_reads,

r.wait\_type,

r.wait\_time,

r.blocking\_session\_id,

r.start\_time,

r.text\_size

FROM

sys.dm\_exec\_sessions s

LEFT JOIN

sys.dm\_exec\_requests r ON s.session\_id = r.session\_id

OUTER APPLY

sys.dm\_exec\_sql\_text(r.sql\_handle) AS r\_sql

WHERE

s.is\_user\_process = 1;

**DMV provides information about transactions and includes details about the transaction, user session, application and the query that initiated it.(Active transaction)**

SELECT

trans.session\_id AS [SESSION ID],

ESes.host\_name AS [HOST NAME],

login\_name AS [Login NAME],

trans.transaction\_id AS [TRANSACTION ID],

tas.name AS [TRANSACTION NAME],

tas.transaction\_begin\_time AS [TRANSACTION BEGIN TIME],

tds.database\_id AS [DATABASE ID],

DBs.name AS [DATABASE NAME]

FROM sys.dm\_tran\_active\_transactions tas

JOIN sys.dm\_tran\_session\_transactions trans ON (trans.transaction\_id = tas.transaction\_id)

LEFT OUTER JOIN sys.dm\_tran\_database\_transactions tds ON (tas.transaction\_id = tds.transaction\_id)

LEFT OUTER JOIN sys.databases AS DBs ON tds.database\_id = DBs.database\_id

LEFT OUTER JOIN sys.dm\_exec\_sessions AS ESes ON trans.session\_id = ESes.session\_id

WHERE ESes.session\_id IS NOT NULL;

**CPU Consumption by Database:**

Select db\_name(pa.database\_id) as [Database\_Name], sum(qs.total\_worker\_time/qs,execution\_count) as AVG\_CPU\_COST

from sys.dm\_exec\_query\_stats qs

cross apply

(select convert(int,value) as [database\_id] from

sys.dm\_exec\_plan\_attributes(qs.plan\_handle)

where attribute=N'dbid) pa

group by [database\_id]

order by AVG\_CPU\_COST DESC

**CPU history:**

select highest\_cpu\_queries.plan\_handle,highest\_cpu\_queries.  
plan\_generation\_num,highest\_cpu\_queries.max\_worker\_time,  
highest\_cpu\_queries.total\_physical\_reads,  
highest\_cpu\_queries.total\_logical\_reads,  
highest\_cpu\_queries.total\_elapsed\_time,q.[text],q.dbid,q.objectid,q.number,q.encrypted,query\_plan  
from (select top 50 qs.plan\_handle,  
qs.plan\_generation\_num,qs.creation\_time, qs.execution\_count, qs.total\_worker\_time,  
qs.max\_worker\_time, qs.total\_elapsed\_time,  
qs.max\_elapsed\_time, qs.total\_logical\_reads, qs.max\_logical\_reads,  
qs.total\_physical\_reads, qs.max\_physical\_reads from sys.dm\_exec\_query\_stats  
qs order by qs.total\_worker\_time DESC)  
as highest\_cpu\_queries  
cross apply sys.dm\_exec\_sql\_text (plan\_handle) as q  
cross apply sys.dm\_exec\_query\_plan (plan\_handle) as qp  
order by highest\_cpu\_queries.total\_worker\_time DESC

<https://blog.sqlauthority.com/2021/09/20/sql-server-troubleshooting-high-cpu/>

𝐇𝐨𝐰 𝐭𝐨 𝐃𝐢𝐚𝐠𝐧𝐨𝐬𝐞 𝐂𝐏𝐔 𝐒𝐩𝐢𝐤𝐞𝐬 𝐢𝐧 𝐒𝐐𝐋 𝐒𝐞𝐫𝐯𝐞𝐫 – 𝐀 𝐑𝐞𝐚𝐥-𝐖𝐨𝐫𝐥𝐝 𝐀𝐩𝐩𝐫𝐨𝐚𝐜𝐡  
  
CPU suddenly hitting 90%+ in your SQL Server and you're unsure why?  
Here’s how I diagnose such CPU spikes step-by-step – based on real-world experience from production environments:  
Real-life scenario:  
A client reported sudden performance degradation. Their SQL Server CPU was constantly above 85%. No recent deployment. No major change. Just a regular weekday – but the system was crawling.  
  
Here’s what I did:  
1. Identify the Culprit Query  
Used this DMV to find high CPU queries:  
SELECT TOP 10   
 qs.total\_worker\_time/qs.execution\_count AS AvgCPU,  
 qs.execution\_count,  
 qs.total\_worker\_time,  
 qs.total\_elapsed\_time,  
 st.text AS QueryText,  
 qp.query\_plan  
FROM sys.dm\_exec\_query\_stats qs  
CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) st  
CROSS APPLY sys.dm\_exec\_query\_plan(qs.plan\_handle)  
ORDER BY AvgCPU DESC  
  
2. Check Wait Stats  
Sometimes it’s not just CPU – it’s what SQL Server is waiting on:  
SELECT TOP 10   
 wait\_type,   
 wait\_time\_ms/1000.0 AS wait\_time\_s,   
 percent\_complete  
FROM sys.dm\_os\_wait\_stats  
ORDER BY wait\_time\_ms DESC  
  
3. Review Execution Plan  
The top query had a missing index and was doing a full table scan on a 20-million-row table.  
  
4. Fix the Root Cause  
After adding a proper index and updating stats, CPU usage dropped from 90% to below 40% – instantly improving response time.

𝗗𝗲𝗮𝗱𝗹𝗼𝗰𝗸𝘀 & 𝗕𝗹𝗼𝗰𝗸𝗶𝗻𝗴 (𝗧-𝗦𝗤𝗟):  
  
Pinpoints blocking chains and helps troubleshoot deadlocks faster.  
SELECT   
 blocking\_session\_id,   
 wait\_type,   
 wait\_time,   
 resource\_description  
FROM sys.dm\_exec\_requests  
WHERE blocking\_session\_id <> 0;  
  
𝗘𝘅𝗽𝗲𝗻𝘀𝗶𝘃𝗲 𝗤𝘂𝗲𝗿𝗶𝗲𝘀 (𝗧-𝗦𝗤𝗟):  
  
Surfaces the queries burning the most resources.  
  
SELECT TOP 10   
 qs.total\_logical\_reads AS Reads,  
 qs.total\_worker\_time/1000 AS CPU\_Time\_ms,  
 qs.execution\_count,  
 SUBSTRING(qt.text,1,200) AS QueryText  
FROM sys.dm\_exec\_query\_stats qs  
CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) qt  
ORDER BY qs.total\_worker\_time DESC;

-----------------------------------------------------------------------------------------------------------------------------------

**Here is the Script to List Database File Latency**

SELECT DB\_NAME(mf.database\_id) AS [Database Name],

mf.physical\_name [Physical Name],

num\_of\_reads, num\_of\_bytes\_read, io\_stall\_read\_ms,

num\_of\_writes, num\_of\_bytes\_written, io\_stall\_write\_ms,

io\_stall, size\_on\_disk\_bytes

FROM sys.dm\_io\_virtual\_file\_stats(null,null) AS vfs

INNER JOIN sys.master\_files AS mf

ON mf.database\_id = vfs.database\_id

and mf.file\_id = vfs.file\_id

-- ORDER BY io\_stall DESC

**Outdated stats:**

select

oject\_name([object\_id]) as "Table",

[sp].[stats\_id] AS "Statistic ID",

[s].[name] AS 'Statistic",

[sp].[last\_updated] AS "Last Updated",

[sp].[rows],

[sp].[rows\_sampled],

[sp].[unfiltered\_rows],

[sp].[modification\_counter] As "Modifications"

from [sys].[stats] AS [s]

outer apply sys.dm\_db\_stats\_properties ([s].[Oject\_id],[s].[stats\_id]) As [sp]

where [s].[oject\_id]=OBJECT\_ID[N'Person.Person2');

/\*Currently Running Session\*/  
  
SELECT   
 SPID      = er.session\_id  
 ,STATUS     = ses.STATUS  
 ,[Login]    = ses.login\_name  
 ,Host      = ses.host\_name  
 ,BlkBy     = er.blocking\_session\_id  
 ,DBName     = DB\_Name(er.database\_id)  
 ,CommandType  = er.command  
 ,ObjectName   = OBJECT\_NAME(st.objectid)  
 ,CPUTime    = er.cpu\_time  
 ,StartTime   = er.start\_time  
 ,TimeElapsed  = CAST(GETDATE() - er.start\_time AS TIME)  
 --,SQLStatement  = st.text  
FROM  sys.dm\_exec\_requests er  
   
 OUTER APPLY sys.dm\_exec\_sql\_text(er.sql\_handle) st  
  
 LEFT JOIN sys.dm\_exec\_sessions ses  
 ON ses.session\_id = er.session\_id  
   
 LEFT JOIN sys.dm\_exec\_connections con  
 ON con.session\_id = ses.session\_id  
  
WHERE  st.text IS NOT NULL

---**Monitor waiting tasks**

select owt.session\_id,owt.exec\_context\_id,owt.wait\_duration\_ms,owt.wait\_type,

owt.blocking\_session\_id,owt.resource\_description

from sys.dm\_os\_waiting\_tasks [owt]

inner join sys.dm\_exec\_sessions [es] on

owt.session\_id=es.session\_id

inner join sys.dm\_exec\_requests [er] on

es.session\_id=er.session\_id

where es.is\_user\_process=1

order by owt.session\_id,owt.exec\_context\_id

Waits:

**SELECT**

    [owt].[session\_id],

    [owt].[exec\_context\_id],

    [owt].[wait\_duration\_ms],

    [owt].[wait\_type],

    [owt].[blocking\_session\_id],

    [owt].[resource\_description],

    CASE [owt].[wait\_type]

**WHEN** N'CXPACKET' **THEN**

            RIGHT ([owt].[resource\_description],

            CHARINDEX (N'=', REVERSE ([owt].[resource\_description])) - 1)

**ELSE** NULL

**END** **AS** [Node ID],

    [es].[program\_name],

    [est].text,

    [er].[database\_id],

    [eqp].[query\_plan],

    [er].[cpu\_time]

**FROM** sys.dm\_os\_waiting\_tasks [owt]

**INNER** JOIN sys.dm\_exec\_sessions [es] **ON**

    [owt].[session\_id] = [es].[session\_id]

**INNER** JOIN sys.dm\_exec\_requests [er] **ON**

    [es].[session\_id] = [er].[session\_id]

OUTER APPLY sys.dm\_exec\_sql\_text ([er].[sql\_handle]) [est]

OUTER APPLY sys.dm\_exec\_query\_plan ([er].[plan\_handle]) [eqp]

**WHERE**

    [es].[is\_user\_process] = 1

**ORDER** **BY**

    [owt].[session\_id],

    [owt].[exec\_context\_id];

GO

**Blocking:**

select

oject\_name([object\_id]) as "Table",

[sp].[stats\_id] AS "Statistic ID",

[s].[name] AS 'Statistic",

[sp].[last\_updated] AS "Last Updated",

[sp].[rows],

[sp].[rows\_sampled],

[sp].[unfiltered\_rows],

[sp].[modification\_counter] As "Modifications"

from [sys].[stats] AS [s]

outer apply sys.dm\_db\_stats\_properties ([s].[Oject\_id],[s].[stats\_id]) As [sp]

where [s].[oject\_id]=OBJECT\_ID[N'Person.Person2');

--Blocking details

if object\_id('tempdb..#blocks') is not null

drop table #blocks

select

spid,

blocked,

replace (replace (st.text, char(10), ' '), char (13), ' ' ) as batch

into

#blocks

from

sys.sysprocesses spr

cross apply

sys.dm\_exec\_sql\_text(spr.sql\_handle) st

go

with blocking\_tree (spid, blocking\_spid, [level], batch)

as

(

select

blc.spid,

blc.blocked,

cast (replicate ('0', 4-len (cast (blc.spid as varchar))) + cast (blc.spid as varchar) as varchar (1000)) as [level],

blc.batch

from

#blocks blc

where

(blc.blocked = 0 or blc.blocked = spid)

and

exists (select \* from #blocks blc2 where blc2.blocked = blc.spid and blc2.blocked <> blc2.spid)

union all

select

blc.spid,

blc.blocked,

cast(bt.[level] + right (cast ((1000 + blc.spid) as varchar (100)), 4) as varchar (1000)) as [level],

blc.batch

from

#blocks as blc

inner join

blocking\_tree bt

on blc.blocked = bt.spid

where

blc.blocked > 0

and

blc.blocked <> blc.spid

)

select

N'' + isnull(replicate (N'| ', len (level)/4 - 2),'') + case when (len(level)/4 - 1) = 0 then '' else '|------ ' end + cast (bt.spid as nvarchar (10)) as blocking\_tree,

spr.lastwaittype AS [type],

spr.loginame AS [login\_name],

db\_name(spr.dbid) as [source\_database],

st.text AS [sql\_text],

case when cur.sql\_handle is null then '' else (select [text] from sys.dm\_exec\_sql\_text (cur.sql\_handle)) end as [cursor\_sql\_text],

db\_name(sli.rsc\_dbid) as [database],

object\_schema\_name(sli.rsc\_objid,sli.rsc\_dbid) as [schema],

object\_name(sli.rsc\_objid, sli.rsc\_dbid) as [table],

spr.waitresource as [wait\_resource],

spr.cmd as [command],

spr.program\_name as [application],

spr.hostname as [hostname],

spr.last\_batch as [last\_batch\_time]

from

blocking\_tree bt

left outer join

sys.sysprocesses spr

on spr.spid = bt.spid

cross apply

sys.dm\_exec\_sql\_text(spr.sql\_handle) st

left join

sys.dm\_exec\_cursors(0) cur

on cur.session\_id = spr.spid

and cur.fetch\_status != 0

join

sys.syslockinfo sli

on sli.req\_spid = spr.spid

and sli.rsc\_type = 5

and object\_name(sli.rsc\_objid, sli.rsc\_dbid) is not null

order by

level asc;

------------------------------------------------------------------------------------------------------------------------------------

WITH cteBL (session\_id, blocking\_these) AS   
(SELECT s.session\_id, blocking\_these = x.blocking\_these FROM sys.dm\_exec\_sessions s   
CROSS APPLY  (SELECT isnull(convert(varchar(6), er.session\_id),'') + ', '    
        FROM sys.dm\_exec\_requests as er  
        WHERE er.blocking\_session\_id = isnull(s.session\_id ,0)  
        AND er.blocking\_session\_id <> 0  
        FOR XML PATH('') ) AS x (blocking\_these)  
)  
SELECT s.session\_id, blocked\_by = r.blocking\_session\_id, bl.blocking\_these  
, batch\_text = t.text, input\_buffer = ib.event\_info, \*   
FROM sys.dm\_exec\_sessions s   
LEFT OUTER JOIN sys.dm\_exec\_requests r on r.session\_id = s.session\_id  
INNER JOIN cteBL as bl on s.session\_id = bl.session\_id  
OUTER APPLY sys.dm\_exec\_sql\_text (r.sql\_handle) t  
OUTER APPLY sys.dm\_exec\_input\_buffer(s.session\_id, NULL) AS ib  
WHERE blocking\_these is not null or r.blocking\_session\_id > 0  
ORDER BY len(bl.blocking\_these) desc, r.blocking\_session\_id desc, r.session\_id;

**Log\_reuse\_wait\_desc is showing ‘replication’ then execute below query:**

EXEC sp\_repldone @xactid = NULL, @xact\_segno = NULL, @numtrans = 0,@time = 0, @reset = 1

Lets again check the log\_reuse\_wait:

SELECT name,recovery\_model\_desc,log\_reuse\_wait\_desc FROM SYS.DATABASES

Query to check connection status of Always on Replica nodes status:

select r.replica\_server\_name, r.endpoint\_url,

rs.connected\_state\_desc, rs.last\_connect\_error\_description,

rs.last\_connect\_error\_number, rs.last\_connect\_error\_timestamp

from sys.dm\_hadr\_availability\_replica\_states rs join sys.availability\_replicas r

on rs.replica\_id=r.replica\_id

where rs.is\_local=1

Last LSN taken:

https://sqlserverblogforum.com/how-to-solve-the-lsn-mismatch-in-sql-server/

DECLARE @db\_name VARCHAR(100)

SELECT @db\_name = 'Employee'

-- Get Backup History

SELECT TOP (30) s.database\_name, s.name, s.description

,m.physical\_device\_name

,CAST(CAST(s.backup\_size / 1000000 AS INT) AS VARCHAR(14)) + ' ' + 'MB' AS bkSize

,CAST(DATEDIFF(second, s.backup\_start\_date, s.backup\_finish\_date) AS VARCHAR(4)) + ' ' + 'Seconds' TimeTaken

,s.backup\_start\_date

,CAST(s.first\_lsn AS VARCHAR(50)) AS first\_lsn

,CAST(s.last\_lsn AS VARCHAR(50)) AS last\_lsn

,CASE s.[type] WHEN 'D'

THEN 'Full'

WHEN 'I'

THEN 'Differential'

WHEN 'L'

THEN 'Transaction Log'

END AS BackupType

,s.server\_name

,s.recovery\_model

FROM msdb.dbo.backupset s

INNER JOIN msdb.dbo.backupmediafamily m ON s.media\_set\_id = m.media\_set\_id

WHERE s.database\_name = @db\_name

ORDER BY backup\_start\_date DESC

,backup\_finish\_date

==**Query to find all tables/objects data file size in kb on a database:**

SELECT

t.object\_id,

OBJECT\_NAME(t.object\_id) ObjectName,

sum(u.total\_pages) \* 8 Total\_Reserved\_kb,

sum(u.used\_pages) \* 8 Used\_Space\_kb,

u.type\_desc,

max(p.rows) RowsCount

FROM

sys.allocation\_units u

JOIN sys.partitions p on u.container\_id = p.hobt\_id

JOIN sys.tables t on p.object\_id = t.object\_id

GROUP BY

t.object\_id,

OBJECT\_NAME(t.object\_id),

u.type\_desc

ORDER BY

Used\_Space\_kb desc,

ObjectName

50844735

SELECT file\_id, name, type\_desc, physical\_name, size, max\_size

FROM sys.database\_files;

**To display data space used, by object and allocation unit, for a database.**

SELECT t.object\_id,

OBJECT\_NAME(t.object\_id) ObjectName,

sum(u.total\_pages) \* 8 Total\_Reserved\_kb,

sum(u.used\_pages) \* 8 Used\_Space\_kb,

u.type\_desc,

max(p.rows) RowsCount

FROM sys.allocation\_units u

JOIN sys.partitions p on u.container\_id = p.hobt\_id

JOIN sys.tables t on p.object\_id = t.object\_id

GROUP BY t.object\_id, OBJECT\_NAME(t.object\_id), u.type\_desc

ORDER BY Used\_Space\_kb desc, ObjectName;

**MANAGE high memory utilization in SQL Server using T-SQL scripts:**

**Memory usage by all the components of memory clerk:**

SELECT

type AS Memory\_Clerk\_Type,

SUM(pages\_kb) / 1024 AS Memory\_Usage\_MB

FROM

sys.dm\_os\_memory\_clerks

GROUP BY

type

ORDER BY

Memory\_Usage\_MB DESC;

-- **Identify most resource-intensive queries**

SELECT TOP 10

qt.text AS 'SQL Text',

qs.total\_worker\_time,

qs.execution\_count,

qs.total\_worker\_time / qs.execution\_count AS avg\_worker\_time,

qs.total\_logical\_reads,

qs.total\_logical\_reads / qs.execution\_count AS avg\_logical\_reads

FROM sys.dm\_exec\_query\_stats qs

CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) qt

ORDER BY qs.total\_worker\_time DESC;

**Buffer Pool Usage:**

SELECT

DB\_NAME(database\_id) AS DatabaseName,

COUNT(\*) \* 8 / 1024 AS BufferPool\_MB

FROM

sys.dm\_os\_buffer\_descriptors

GROUP BY

database\_id

ORDER BY

BufferPool\_MB DESC;

**Total memory consumption details query:**

SELECT

physical\_memory\_in\_use\_kb / 1024 AS Physical\_Memory\_Used\_MB,

large\_page\_allocations\_kb / 1024 AS Large\_Page\_Allocations\_MB,

locked\_page\_allocations\_kb / 1024 AS Locked\_Page\_Allocations\_MB,

total\_virtual\_address\_space\_kb / 1024 AS Total\_Virtual\_Address\_Space\_MB,

virtual\_address\_space\_reserved\_kb / 1024 AS Virtual\_Address\_Space\_Reserved\_MB,

virtual\_address\_space\_committed\_kb / 1024 AS Virtual\_Address\_Space\_Committed\_MB,

virtual\_address\_space\_available\_kb / 1024 AS Virtual\_Address\_Space\_Available\_MB,

page\_fault\_count,

memory\_utilization\_percentage

FROM

sys.dm\_os\_process\_memory;  
  
1. Check Current Memory Configuration:  
  
SELECT   
  name, value, value\_in\_use  
FROM sys.configurations  
WHERE name IN ('min server memory (MB)', 'max server memory (MB)');  
   
2. Identify Memory-Intensive Queries :  
   
SELECT TOP 10   
  total\_worker\_time/execution\_count AS avg\_cpu\_time,  
  total\_elapsed\_time/execution\_count AS avg\_elapsed\_time,  
  total\_logical\_reads/execution\_count AS avg\_logical\_reads,  
  total\_logical\_writes/execution\_count AS avg\_logical\_writes,  
  execution\_count,  
  statement\_text  
FROM sys.dm\_exec\_query\_stats   
CROSS APPLY sys.dm\_exec\_sql\_text(sql\_handle)  
ORDER BY avg\_logical\_reads DESC;  
  
  
3. Monitor Memory-Related Performance Counters :  
  
SELECT   
  object\_name,  
  counter\_name,  
  cntr\_value  
FROM sys.dm\_os\_performance\_counters  
WHERE counter\_name IN ('Memory Grants Pending', 'Target Server Memory (KB)', 'Total Server Memory (KB)', 'Page Life Expectancy');  
--or you may use   
WHERE object\_name LIKE '%Memory%';  
   
4. Optimize Memory-Intensive Queries :  
  - Add appropriate indexes to tables involved in memory-intensive queries.  
  - Rewrite complex queries to improve efficiency.  
  - Update statistics for tables involved in queries.  
-UPDATE STATISTICS TableName;  
  
5. Set Max Server Memory :  
     
  -- Set max server memory to 560 GB (for example)  
  EXEC sp\_configure 'max server memory', 16384; -- in MB  
  RECONFIGURE;  
     
6. Check Memory-Optimized Tables :  
     
  SELECT   
    object\_name,  
    memory\_allocated\_for\_table\_kb AS 'Memory Allocated (KB)',  
    memory\_used\_by\_table\_kb AS 'Memory Used (KB)'  
  FROM sys.dm\_db\_xtp\_table\_memory\_stats;

**Outdated stats:**

select

oject\_name([object\_id]) as "Table",

[sp].[stats\_id] AS "Statistic ID",

[s].[name] AS 'Statistic",

[sp].[last\_updated] AS "Last Updated",

[sp].[rows],

[sp].[rows\_sampled],

[sp].[unfiltered\_rows],

[sp].[modification\_counter] As "Modifications"

from [sys].[stats] AS [s]

outer apply sys.dm\_db\_stats\_properties ([s].[Oject\_id],[s].[stats\_id]) As [sp]

where [s].[oject\_id]=OBJECT\_ID[N'Person.Person2');

**Blocking:**

select

oject\_name([object\_id]) as "Table",

[sp].[stats\_id] AS "Statistic ID",

[s].[name] AS 'Statistic",

[sp].[last\_updated] AS "Last Updated",

[sp].[rows],

[sp].[rows\_sampled],

[sp].[unfiltered\_rows],

[sp].[modification\_counter] As "Modifications"

from [sys].[stats] AS [s]

outer apply sys.dm\_db\_stats\_properties ([s].[Oject\_id],[s].[stats\_id]) As [sp]

where [s].[oject\_id]=OBJECT\_ID[N'Person.Person2');

select qs.session\_id,qs.status,qs.command,qs.wait\_type,qs.wait\_resource,qs.reads,qs.database\_id,qs.blocking\_session\_id,

qs.writes,dmsqlt.text from sys.dm\_exec\_requests as qs CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) as dmsqlt

where qs.status not in ('background','Sleeping')

SELECT \*

FROM sys.dm\_tran\_locks

WHERE resource\_database\_id = (SELECT db.database\_id

FROM sys.databases AS db

WHERE name = 'SJINT4');

- -Returns information about each request that is executing in SQL Server

SELECT \*

FROM sys.dm\_exec\_requests

GO

-- Returns information about the request that is executing in SQL Server where session\_id is greater than 50

SELECT \*

FROM sys.dm\_exec\_requests

WHERE session\_id > 50

GO

-- Returns information about the request that is executing in SQL Server.

-- SQL Handle: It is a hash of the SQL Text containing all your formatting (casing, spaces, etc).

SELECT \*

FROM sys.dm\_exec\_requests er

CROSS APPLY sys.dm\_exec\_sql\_text(sql\_handle) GO

Here is another script where the result will be below the table.

SELECT

qs.Session\_ID,

Blocking\_Session\_ID, qs.Status,

Wait\_Type, Wait\_Time, Wait\_Resource, SUBSTRING(st.text, (qs.statement\_start\_offset/2)+1,

((CASE qs.statement\_end\_offset

WHEN -1 THEN DATALENGTH(st.text)

ELSE qs.statement\_end\_offset

END - qs.statement\_start\_offset)/2) + 1) AS statement\_text, GetDate() SnapshotDateTime,

--Open\_Transaction\_Count,

ss.PROGRAM\_NAME, ss.HOST\_NAME, ss.Login\_Name

FROM sys.dm\_exec\_requests AS qs INNER JOIN sys.dm\_exec\_sessions ss

ON qs.session\_id = ss.session\_id

CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) AS st

WHERE Wait\_Time > 0

ORDER BY Wait\_Time DESC

Or

SELECT

qs.Session\_ID,

Blocking\_Session\_ID,

qs.Status,

Wait\_Type,

Wait\_Time,

Wait\_Resource,

SUBSTRING(st.text, (qs.statement\_start\_offset/2)+1,

((CASE qs.statement\_end\_offset

WHEN -1 THEN DATALENGTH(st.text)

ELSE qs.statement\_end\_offset

END - qs.statement\_start\_offset)/2) + 1) AS statement\_text,

GetDate() AS SnapshotDateTime,

*--Open\_Transaction\_Count,*

ss.PROGRAM\_NAME,

ss.HOST\_NAME,

ss.Login\_Name

FROM

sys.dm\_exec\_requests AS qs

INNER JOIN

sys.dm\_exec\_sessions ss ON qs.session\_id = ss.session\_id

CROSS APPLY

sys.dm\_exec\_sql\_text(qs.sql\_handle) AS st

WHERE

Wait\_Time > 0

ORDER BY

Wait\_Time DESC

Kill blocking process

1) Use an activity monitor to kill the process

Right-click on the SQL instance name -> Activity monitor -> Expand the process section -> Find the relevant process id and right-click to kill the process.

2) Run the kill command to kill the process

KILL PROCESS\_ID

🔧 \*\*Resolving SQL Server Memory Issues\*\* 🔧  
  
Experiencing memory issues in SQL Server? Here's how to diagnose and fix them:  
  
### \*\*Symptoms:\*\*  
- Slow queries  
- High CPU usage  
- Frequent timeouts  
  
  
### \*\*Solutions:\*\*  
  
1. \*\*Identify High-Consumption Queries:\*\*  
 ```sql  
 SELECT TOP 10   
 [Total Memory (MB)] = (w.total\_memory\_kb/1024),  
 [Query Text] = t.text  
 FROM sys.dm\_exec\_query\_stats AS qs  
 CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) AS t  
 CROSS APPLY sys.dm\_exec\_query\_memory\_grants AS w  
 ORDER BY w.total\_memory\_kb DESC;  
 ```  
  
2. \*\*Optimize Queries:\*\* Refactor inefficient queries and add indexes.  
  
3. \*\*Adjust Memory Settings:\*\*  
 ```sql  
 EXEC sp\_configure 'max server memory (MB)', 8192; -- Set max to 8GB  
 RECONFIGURE;  
 EXEC sp\_configure 'min server memory (MB)', 4096; -- Set min to 4GB  
 RECONFIGURE;  
 ```  
  
4. \*\*Monitor and Tune Indexes:\*\* Update statistics and rebuild indexes.  
 ```sql  
 EXEC sp\_updatestats;  
 ALTER INDEX ALL ON [TableName] REBUILD;  
 ```  
  
5. \*\*Check for Memory Leaks:\*\*  
 ```sql  
 SELECT type, SUM(pages\_kb) / 1024 AS memory\_used\_mb   
 FROM sys.dm\_os\_memory\_clerks   
 GROUP BY type   
 ORDER BY memory\_used\_mb DESC;

𝗧-𝗦𝗤𝗟 𝗤𝘂𝗲𝗿𝘆: 𝗗𝗶𝘀𝗸 𝗨𝘀𝗮𝗴𝗲 𝗯𝘆 𝗗𝗮𝘁𝗮𝗯𝗮𝘀𝗲

-- Database file space usage

SELECT

DB\_NAME(database\_id) AS DatabaseName,

file\_id,

type\_desc AS FileType,

size/128.0 AS TotalSizeMB,

size/128.0 - CAST(FILEPROPERTY(name, 'SpaceUsed') AS INT)/128.0 AS FreeSpaceMB

FROM sys.master\_files

ORDER BY FreeSpaceMB ASC;

💻 𝗣𝗼𝘄𝗲𝗿𝗦𝗵𝗲𝗹𝗹 𝗦𝗰𝗿𝗶𝗽𝘁 𝗳𝗼𝗿 𝗗𝗿𝗶𝘃𝗲 𝗛𝗲𝗮𝗹𝘁𝗵

Get-PSDrive -PSProvider FileSystem |

Select-Object Name, @{Name="Used(GB)";Expression={[math]::Round($\_.Used/1GB,2)}},

@{Name="Free(GB)";Expression={[math]::Round($\_.Free/1GB,2)}},

@{Name="Capacity(GB)";Expression={[math]::Round($\_.Used/1GB + $\_.Free/1GB,2)}}

🔍 Run these together, and you’ll know exactly which drives are running low and which databases need attention.

---------------------------------------------------------------------------------------------------

**Script to check connection status of replicas from primary:**

select \* replica\_server\_name,r.endpoint\_url,rs\_connected\_state\_desc,

rs\_last\_connect\_error\_description,rs.last\_connect\_error\_number,rs\_last\_connect\_error\_timestamp

from sys.dm\_hadr\_availability\_replica\_states rs join sys.availability\_replicas r

on rs.replica\_id=r.replica\_id

where rs.is\_local<1

--**create a report of SQL Server Agent jobs, focusing on their status, run dates, and durations.**

set nocount on

CREATE TABLE #tJobReport

(

lngID INTEGER IDENTITY(1,1)

,jobname VARCHAR(80)

,jobEnabled varchar(3)

,status VARCHAR(10)

,rundate VARCHAR(10)

,runtime CHAR(8)

,runduration CHAR(8)

)

INSERT INTO #tJobReport (jobname, jobEnabled, status, rundate, runtime, runduration)

SELECT [sj.name](http://sj.name),

Case sj.enabled When 1 Then 'Yes' Else 'No' End as jobEnabled,

CASE sjh.run\_status

WHEN 0 THEN 'Failed'

WHEN 1 THEN 'Succeeded'

WHEN 2 THEN 'Retry'

WHEN 3 THEN 'Canceled'

WHEN 4 THEN 'Executing'

ELSE 'Unknown'

END,

--Convert Integer date to regular datetime

SUBSTRING(CAST(sjh.run\_date AS CHAR(8)),5,2) + '/' +

RIGHT(CAST(sjh.run\_date AS CHAR(8)),2) + '/' +

LEFT(CAST(sjh.run\_date AS CHAR(8)),4)

--Change run time into hh:mm:ss

, LEFT(RIGHT('000000' + CAST(run\_time AS VARCHAR(10)),6),2) + ':' +

SUBSTRING(RIGHT('000000' + CAST(run\_time AS VARCHAR(10)),6),3,2) + ':' +

RIGHT(RIGHT('000000' + CAST(run\_time AS VARCHAR(10)),6),2)

--Change run duration into hh:mm:ss

, LEFT(RIGHT('000000' + CAST(run\_duration AS VARCHAR(10)),6),2) + ':' +

SUBSTRING(RIGHT('000000' + CAST(run\_duration AS VARCHAR(10)),6),3,2) + ':' +

RIGHT(RIGHT('000000' + CAST(run\_duration AS VARCHAR(10)),6),2)

FROM msdb.dbo.sysjobs sj --job id and name

--Job history

INNER JOIN msdb.dbo.sysjobhistory sjh

ON sj.job\_id = sjh.job\_id

--Join for new history rows

left JOIN #tJobReport jr

ON [sj.name](http://sj.name) = jr.jobname

AND SUBSTRING(CAST(sjh.run\_date AS CHAR(8)),5,2) + '/' +

RIGHT(CAST(sjh.run\_date AS CHAR(8)),2) + '/' +

LEFT(CAST(sjh.run\_date AS CHAR(8)),4) = jr.rundate

AND LEFT(RIGHT('000000' + CAST(run\_time AS VARCHAR(10)),6),2) + ':' +

SUBSTRING(RIGHT('000000' + CAST(run\_time AS VARCHAR(10)),6),3,2) + ':' +

RIGHT(RIGHT('000000' + CAST(run\_time AS VARCHAR(10)),6),2) = jr.runtime

--Only job outcome not each step outcome

WHERE sjh.step\_id = 0

--Only new data

AND jr.lngID IS NULL

ORDER BY sjh.run\_date

select jobname, max(lngID) as lngID

into #temp2

from #tJobReport

group by jobname

-- Verify Jobs in running status

CREATE TABLE #Temp3         (job\_id                UNIQUEIDENTIFIER NOT NULL,

                            last\_run\_date         INT              NOT NULL,

                            last\_run\_time         INT              NOT NULL,

                            next\_run\_date         INT              NOT NULL,

                            next\_run\_time         INT              NOT NULL,

                            next\_run\_schedule\_id  INT              NOT NULL,

                            requested\_to\_run      INT              NOT NULL, -- BOOL

                            request\_source        INT              NOT NULL,

                            request\_source\_id     sysname          COLLATE database\_default NULL,

                            running               INT              NOT NULL, -- BOOL

                            current\_step          INT              NOT NULL,

                            current\_retry\_attempt INT              NOT NULL,

                            job\_state             INT              NOT NULL)

      INSERT #Temp3

      EXECUTE master.dbo.xp\_sqlagent\_enum\_jobs 1, 'dbo'

  DELETE #Temp3

      WHERE running = 1

--select ltrim(rtrim(a.jobname)) as jobname,a.jobEnabled, a.status, a.rundate, left(a.runtime,5) as runtime, a.runduration

select count(1) as Qty\_Failed\_Jobs

from #tJobReport a

JOIN #Temp2 b on a.jobname = b.jobname and a.lngID = b.lngID

JOIN msdb.dbo.sysjobs\_view c on b.jobname = [c.name](http://c.name)

LEFT JOIN #Temp3 d on c.job\_id = d.job\_id

where (a.status = 'Failed' or a.status = 'Canceled')

   AND a.jobEnabled = 'Yes'

   AND a.rundate >= convert(varchar,dateadd(dd,-1,getdate()),101)

drop table #tJobReport

drop table #Temp2

drop table #Temp3

set nocount off

query to find orphaned users :  
  
SELECT dp.type\_desc, dp.sid, [dp.name](http://dp.name/) AS user\_name   
FROM sys.database\_principals AS dp   
LEFT JOIN sys.server\_principals AS sp   
 ON dp.sid = sp.sid   
WHERE sp.sid IS NULL   
 AND dp.authentication\_type\_desc = 'INSTANCE'

**Always on**

**--AG Status --lIKE Listener/Listerner\_Status/AG\_Name/AG\_Role/Seeding\_mode/Failove\_Mode/Health\_status/ListnerIP/Availability\_Mode/Read\_Only Destination/**

[](https://blogger.googleusercontent.com/img/b/R29vZ2xl/AVvXsEikSDl5l3CWbFTKc06iVntMKOlLBOiN_cvt45K4yxUt4Pcza6M9ok7OPaG8ZG_NFP9rSO4kGAsHZ4_fIDpcmx371i-ShahkdyUQHpkcstuYhahXib_-MonpRFZHNK7zpP0BjG9AzW1qb2S5XwFd9qSiGdRO4GtC_sx2mH9vvYBp1qlQldQ-1IAvfLDeaeU/s1713/AG.png)

select ag.name as "Availability Group"

 , ar.replica\_server\_name as "Source Replica"

 , ar2.replica\_server\_name as "Read-Only Destination"

 , ars.role\_desc as "AG\_Role"

 , al.dns\_name "Listener"

 , ar.failover\_mode\_desc "Failover\_Mode"

 , ar.seeding\_mode\_desc

 , rl.routing\_priority as "Routing Priority"

 , ar.secondary\_role\_allow\_connections\_desc as "Allowed Secondary Role"

 , ar2.read\_only\_routing\_url as "Read-Only Routing Url"

 , ah.synchronization\_health\_desc "HealthStatus"

 , agl.ip\_address "ListenerIPaddress"

 , state\_desc "Listener\_Status"

 , ar.availability\_mode\_desc "Availability\_Mode"

from sys.availability\_read\_only\_routing\_lists rl

 inner join sys.availability\_replicas ar on rl.replica\_id = ar.replica\_id

 inner join sys.availability\_replicas ar2 on rl.read\_only\_replica\_id = ar2.replica\_id

 inner join sys.dm\_hadr\_availability\_replica\_states ars on ars.replica\_id=ar.replica\_id

 inner join sys.availability\_groups ag on ar.group\_id = ag.group\_id

 inner join sys.dm\_hadr\_availability\_group\_states ah on ah.group\_id = ag.group\_id

 inner join sys.availability\_group\_listeners al on al.group\_id = ag.group\_id

 inner join sys.availability\_group\_listener\_ip\_addresses agl on agl.listener\_id=al.listener\_id

order by ag.name, ar.replica\_server\_name, rl.routing\_priority

**Useful DMV queries for SQL Server Performance Troubleshooting**

Below are some useful DMV queries to help identify and troubleshoot performance bottlenecks.

**Identify CPU-Intensive Queries**

SELECT TOP 10

qs.sql\_handle,

qs.execution\_count,

qs.total\_worker\_time AS TotalCPU,

qs.total\_worker\_time / qs.execution\_count AS AvgCPU,

SUBSTRING(qt.text, qs.statement\_start\_offset / 2,

(CASE

WHEN qs.statement\_end\_offset = -1 THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) \* 2

ELSE qs.statement\_end\_offset

END - qs.statement\_start\_offset) / 2) AS query\_text

FROM

sys.dm\_exec\_query\_stats AS qs

CROSS APPLY

sys.dm\_exec\_sql\_text(qs.sql\_handle) AS qt

ORDER BY

TotalCPU DESC;

**Identify Queries with High Logical Reads**

SELECT TOP 10

qs.sql\_handle,

qs.execution\_count,

qs.total\_logical\_reads AS TotalReads,

qs.total\_logical\_reads / qs.execution\_count AS AvgReads,

SUBSTRING(qt.text, qs.statement\_start\_offset / 2,

(CASE

WHEN qs.statement\_end\_offset = -1 THEN LEN(CONVERT(NVARCHAR(MAX), qt.text)) \* 2

ELSE qs.statement\_end\_offset

END - qs.statement\_start\_offset) / 2) AS query\_text

FROM

sys.dm\_exec\_query\_stats AS qs

CROSS APPLY

sys.dm\_exec\_sql\_text(qs.sql\_handle) AS qt

ORDER BY

TotalReads DESC;

**Find Missing Indexes**

SELECT

migs.avg\_total\_user\_cost \* (migs.avg\_user\_impact / 100.0) \* (migs.user\_seeks + migs.user\_scans) AS improvement\_measure,

mid.statement AS TableName,

mid.equality\_columns,

mid.inequality\_columns,

mid.included\_columns

FROM

sys.dm\_db\_missing\_index\_group\_stats AS migs

INNER JOIN

sys.dm\_db\_missing\_index\_groups AS mig ON migs.group\_handle = mig.index\_group\_handle

INNER JOIN

sys.dm\_db\_missing\_index\_details AS mid ON mig.index\_handle = mid.index\_handle

ORDER BY

improvement\_measure DESC;

**Monitor Wait Statistics**

SELECT

wait\_type,

waiting\_tasks\_count,

wait\_time\_ms / 1000.0 AS wait\_time\_seconds,

(wait\_time\_ms - signal\_wait\_time\_ms) / 1000.0 AS resource\_wait\_time\_seconds,

signal\_wait\_time\_ms / 1000.0 AS signal\_wait\_time\_seconds

FROM

sys.dm\_os\_wait\_stats

ORDER BY

wait\_time\_ms DESC;

**Check Index Usage Statistics**

SELECT

OBJECT\_NAME(s.object\_id) AS TableName,

i.name AS IndexName,

i.index\_id,

s.user\_seeks,

s.user\_scans,

s.user\_lookups,

s.user\_updates

FROM

sys.dm\_db\_index\_usage\_stats AS s

INNER JOIN

sys.indexes AS i ON s.object\_id = i.object\_id AND i.index\_id = s.index\_id

WHERE

s.database\_id = DB\_ID('YourDatabaseName')

ORDER BY

s.user\_seeks DESC;

**Analyze Query Performance**

SELECT

TOP 10

qs.sql\_handle,

qs.plan\_handle,

qs.execution\_count,

qs.total\_elapsed\_time / 1000 AS total\_elapsed\_time\_ms,

qs.total\_worker\_time / 1000 AS total\_worker\_time\_ms,

qs.total\_logical\_reads,

qs.total\_logical\_writes,

qs.total\_physical\_reads,

qs.creation\_time,

qs.last\_execution\_time,

SUBSTRING(st.text, (qs.statement\_start\_offset / 2) + 1,

((CASE qs.statement\_end\_offset

WHEN -1 THEN DATALENGTH(st.text)

ELSE qs.statement\_end\_offset

END - qs.statement\_start\_offset) / 2) + 1) AS statement\_text,

DB\_NAME(st.dbid) AS database\_name

FROM

sys.dm\_exec\_query\_stats AS qs

CROSS APPLY

sys.dm\_exec\_sql\_text(qs.sql\_handle) AS st

ORDER BY

qs.total\_elapsed\_time DESC;

**Identify Blocking Sessions**

SELECT

blocking\_session\_id AS BlockingSessionID,

session\_id AS BlockedSessionID,

wait\_type,

wait\_time / 1000 AS wait\_time\_seconds,

wait\_resource,

SUBSTRING(st.text, (r.statement\_start\_offset / 2) + 1,

((CASE r.statement\_end\_offset

WHEN -1 THEN DATALENGTH(st.text)

ELSE r.statement\_end\_offset

END - r.statement\_start\_offset) / 2) + 1) AS query\_text,

DB\_NAME(r.database\_id) AS database\_name

FROM

sys.dm\_exec\_requests AS r

INNER JOIN

sys.dm\_exec\_sessions AS s ON r.session\_id = s.session\_id

CROSS APPLY

sys.dm\_exec\_sql\_text(r.sql\_handle) AS st

WHERE

blocking\_session\_id <> 0

ORDER BY

wait\_time\_seconds DESC;

**Inspect TempDB Usage**

SELECT

SUM(user\_objects\_alloc\_page\_count) AS UserObjectsPagesUsed,

SUM(internal\_objects\_alloc\_page\_count) AS InternalObjectsPagesUsed,

SUM(version\_store\_alloc\_page\_count) AS VersionStorePagesUsed,

SUM(unallocated\_extent\_page\_count) AS UnallocatedPages,

SUM(mixed\_extent\_page\_count) AS MixedPages

FROM

sys.dm\_db\_file\_space\_usage;

**Analyze Active Expensive Queries**

SELECT

r.session\_id,

r.cpu\_time,

r.total\_elapsed\_time,

r.reads,

r.writes,

r.logical\_reads,

SUBSTRING(qt.text, (r.statement\_start\_offset / 2) + 1,

((CASE r.statement\_end\_offset

WHEN -1 THEN DATALENGTH(qt.text)

ELSE r.statement\_end\_offset

END - r.statement\_start\_offset) / 2) + 1) AS query\_text,

DB\_NAME(r.database\_id) AS database\_name,

r.status,

r.start\_time,

r.command

FROM

sys.dm\_exec\_requests AS r

CROSS APPLY

sys.dm\_exec\_sql\_text(r.sql\_handle) AS qt

ORDER BY

r.cpu\_time DESC;

**Identify Fragmented Indexes**

SELECT

OBJECT\_NAME(ips.object\_id) AS TableName,

i.name AS IndexName,

ips.index\_id,

ips.avg\_fragmentation\_in\_percent,

ips.page\_count

FROM

sys.dm\_db\_index\_physical\_stats(DB\_ID(), NULL, NULL, NULL, 'DETAILED') AS ips

INNER JOIN

sys.indexes AS i ON ips.object\_id = i.object\_id AND ips.index\_id = i.index\_id

WHERE

ips.avg\_fragmentation\_in\_percent > 10 -- Adjust the threshold as needed

AND ips.page\_count > 1000 -- Adjust the threshold as needed

ORDER BY

ips.avg\_fragmentation\_in\_percent DESC;

[**1-Always on AG Health Check Report**](https://sqldbachamps.wordpress.com/2022/09/11/1-always-on-ag-health-check-report/)

**Always on AG Health Check Report -- Author: Dinesh Kumar**

--https://www.sqlservercentral.com/scripts/always-on-ag-dashboard-report

============================================================================================================

T-SQL script which will trigger the Dashboard report whenever there is an issue with any of the node either Primary or Secondary.

To schedule this script, Place it in SQL agent job.

============================================================================================================

This job should be configured on all primary and secondary nodes.

This job will run on all nodes and check whether there is any issue with any of the database(s) or node(s) on all primary and secondary nodes.

If it founds anything wrong like database is in “not synchronizing” state, in such cases, this job will trigger the email to recipient(s).

This email will carry the AG Dashboard report which will tell us about the status of all database(s) and node(s) belongs to the different AG’s.

============================================================================================================

**This script is divided in 2 different parts.**

First part will check whether the current node is Primary,

If current node is primary then execution will go to the second part

otherwise it will complete without generating any report.

In my environment, I have configured a 2 steps(separate step for each part) job which runs in every 15 mins.

============================================================================================================

**First script:**

**============**

use master

go

Declare @databasename varchar(250)

SET @databasename =(Select DISTINCT Top 1 d.name from sys.databases d

INNER JOIN master.sys.dm\_hadr\_database\_replica\_states drs

on d.replica\_id = drs.replica\_id)

if sys.fn\_hadr\_is\_primary\_replica (@databasename) <> 1

Begin

RAISERROR('This is not preferred primary replica',1,1)

END

**Second scrpt:**

**============**

USE master

GO

DECLARE @DB\_count int

DECLARE @tableHTML NVARCHAR(MAX) ;

SET @tableHTML =

N'<p> Hi Team </p>'+

N'<p> Some of the node(s) is/are not healthy. Please use following Database Availability report to find node/database name.</>'+

N'<H1> AG Databases Availability Report</H1>' +

N'<table border="1">' +

N'<tr><th> Availability Group </th><th> Server Name </th><th> Database Name </th>' +

N'<th> DB Synchronization State </th><th> Node health desc </th><th> log reuse wait desc </th><th> Last Commit time </th> </tr>' +

CAST ( ( SELECT td = ag.name, ' ',

td = arcs.replica\_server\_name,' ',

td = d.name,' ',

td = drs.synchronization\_state\_desc, ' ',

td = drs.synchronization\_health\_desc, ' ',

td = d.log\_reuse\_wait\_desc,' ',

td = Convert(nvarchar,drs.last\_commit\_time,20)

FROM master.sys.databases d

INNER JOIN master.sys.dm\_hadr\_database\_replica\_states drs

ON d.database\_id=drs.database\_id

INNER JOIN master.sys.dm\_hadr\_availability\_replica\_cluster\_states arcs

ON drs.replica\_id=arcs.replica\_id

INNER JOIN master.sys.availability\_groups ag

ON ag.group\_id=drs.group\_id

ORDER BY arcs.replica\_server\_name ASC

FOR XML PATH('tr'), TYPE

) AS NVARCHAR(MAX) ) +

N'</table>' ;

Set @DB\_count = (Select count(drs.synchronization\_health\_desc)

FROM master.sys.databases d

INNER JOIN master.sys.dm\_hadr\_database\_replica\_states drs

ON d.database\_id=drs.database\_id

INNER JOIN master.sys.dm\_hadr\_availability\_replica\_cluster\_states arcs

ON drs.replica\_id=arcs.replica\_id

**where drs.synchronization\_health\_desc NOT LIKE 'HEALTHY' OR drs.synchronization\_state\_desc NOT IN ('SYNCHRONIZED','SYNCHRONIZING'))**

IF @DB\_count > 0

EXEC msdb.dbo.sp\_send\_dbmail

@profile\_name='**SQL Server Profile Name**', -- Add here your respective DB Mail Profile.

@recipients= '**Recipients email id**', -- Add here required email id's.

@importance = 'High',

@subject = '**CRITICAL-Node(s) is/are unhealthy**',

@body = @tableHTML,

@body\_format = 'HTML';

else

Print '**All nodes are healthy**'

GO

**Check the last date for rebuild the index in SQL Server**

USE AdventureWorks

GO

SELECT name AS index\_name,

STATS\_DATE(OBJECT\_ID, index\_id) AS StatsUpdated

FROM sys.indexes

WHERE OBJECT\_ID = OBJECT\_ID('HumanResources.Employees')

-- **See how many pages your table uses**

SELECT

OBJECT\_NAME(ps.object\_id) AS TableName,

ps.index\_id,

ps.index\_type\_desc,

ps.page\_count,

ps.avg\_page\_space\_used\_in\_percent

FROM sys.dm\_db\_index\_physical\_stats(DB\_ID(), NULL, NULL, NULL, 'DETAILED') ps

ORDER BY ps.page\_count DESC;

**SQL Server Distributed availability group(DAG). Questions...**  
  
DAG :  
1.) What is DAG and why it is used?  
2.) How does a DAG AG different from a traditional Availability Group?  
3.) what are the main cases of DAG AG's?  
4.) what are the prerequisites for setting up a DAG?  
5.) How does quorum management work in Availability Group's?  
6.) what are the steps to configure a Distributed Availability Group's?  
7.) How do distributed Availability Group handle failover scenarios?  
8.) what are the data synchronization options for Distributed Availability Group's?  
9.) How do you monitor a distributed Availability Group in SQL Server?  
10.)what are the common issue with distributed Availability Group? How do you trouble shoot them?  
11.)can Distributed Availability Group be used in hybrid cloud environments?  
12.) what are the limitations of distributed Availability Group?  
13.) how would you use a distributed Availability Group for database migration?  
14.) what are the best practices to minimize downtime during a distributed Availability Group?  
15.) How do backups works in a Distributed Availability Group?  
16.) how do backups work in distributed Availability Group environments?  
17.) how do distributed Availability Group ensure data consistency between replicas?

--Query running slow how to troubleshoot the sql server.

 Step 1: Immediate Triage: Global or Specific?  
  
✅ Check if slowness is server wide or a few queries  
Find top resource consumers:  
  
SELECT TOP 10 total\_elapsed\_time/1000 AS ElapsedMS,  
 execution\_count,  
 total\_logical\_reads,  
 (SELECT TEXT FROM sys.dm\_exec\_sql\_text(sql\_handle)) AS Query  
FROM sys.dm\_exec\_query\_stats  
ORDER BY total\_elapsed\_time DESC;  
  
📈 Step 2: Live Monitoring: What’s Running Now?  
Use Activity Monitor, sp\_whoisactive, or:  
  
SELECT \* FROM sys.dm\_exec\_requests WHERE status = 'running';  
Watch for:  
Blocking  
Wait types (sys.dm\_os\_waiting\_tasks)  
TempDB usage  
CPU pressure  
  
⏳ Step 3: Wait Stats: The Bottleneck Clues  
Get current waits:  
  
SELECT wait\_type, wait\_time\_ms  
FROM sys.dm\_os\_wait\_stats  
ORDER BY wait\_time\_ms DESC;  
Patterns to note:  
PAGEIOLATCH\_\* = Disk I/O  
CXPACKET = Parallelism  
LCK\_\* = Locking  
SOS\_SCHEDULER\_YIELD = CPU  
  
🧠 Step 4: Query Plans: Right Plan or Sniffed?  
Use Query Store or:  
  
SELECT qp.query\_plan qt.text  
FROM sys.dm\_exec\_query\_stats qs  
CROSS APPLY sys.dm\_exec\_query\_plan(qs.plan\_handle) qp  
CROSS APPLY sys.dm\_exec\_sql\_text(qs.sql\_handle) qt  
Look for:  
Parameter sniffing  
Stats issues  
Index scans  
Row estimation errors  
  
🛠️ Step 5: Indexes & Stats: Healthy or Not?  
Check index fragmentation:  
  
SELECT \* FROM sys.dm\_db\_index\_physical\_stats(DB\_ID(), NULL, NULL, NULL, 'SAMPLED');  
Rebuild/reorganize as needed  
Update stats:  
EXEC sp\_updatestats;  
  
⚙️ Step 6: Blocking & Deadlocks  
Run:  
EXEC sp\_who2;  
SELECT \* FROM sys.dm\_tran\_locks;  
Capture deadlocks via Extended Events  
Kill blockers carefully  
  
🧪 Step 7: TempDB & I/O Checks  
Monitor TempDB:  
SELECT \* FROM sys.dm\_db\_file\_space\_usage;  
Check disk latency:  
sys.dm\_io\_virtual\_file\_stats  
Confirm enough TempDB files  
  
🔄 Step 8: Recent Changes  
Ask:  
Any recent deployments?  
Stats updated?  
Indexes dropped?  
  
🛡️ Step 9: Quick Mitigations  
Use RECOMPILE for sniffing  
Add missing indexes (evaluate impact!)  
Restart heavy agent jobs  
Kill blockers (with caution)  
  
🧰 Step 10: Prevent Future Issues  
Enable Query Store  
Use Extended Events  
Set up alerts & monitoring  
Automate index/stats jobs  
  
✅ Final Thought:  
 Be data-driven. Don’t guess observe, measure, act. Know your performance baseline so you can compare and correct.

```