Course hands on lab content <https://www.dropbox.com/s/x74iqdqevr2eknw/SQL_Admin_JumpStart.iso?dl=0>

Hands on content at <https://www.microsoft.com/en-au/learning/companion-moc.aspx>

Mva hands on lab content <https://www.dropbox.com/s/x74iqdqevr2eknw/SQL_Admin_JumpStart.iso?dl=0>

* 1GiB = 1024MiB = 1024x1024KiB = 1024x1024x1024B
* 1GB = 1000MB = 1000x1000KB = 1000x1000x1000B

CRYSTALDISKMARK AND DISKSPD

--find the open connections to the database

USE master

SELECT \* FROM sys.sysprocesses WHERE dbid = DB\_ID('TinyDB')

--this should also give some idea

exec sp\_who2

exec sp\_whoisactive

KILL <SPID>

--you could also try to set it to read-only in which case it will ask u if u want to close all connections

--u can also open Activity monitor in SSMS context menu for the server to find the processes

--get the current instance configs

USE master;

GO

SELECT \* FROM sys.configurations;

GO

-- Step 1: Open and execute the query 91 - Workload.sql from

-- Solution Explorer. WARNING: Make sure you have followed

-- the instructions at the top of that window before clicking

-- Execute.

-- Step 2: Query the currently executing requests.

-- Note that a large number of requests is returned but that

-- most are system requests.

SELECT \* FROM sys.dm\_exec\_requests;

GO

-- Step 3: The is\_user\_process column of the sys.dm\_exec\_sessions view can

-- be used to filter out system activity

SELECT \* FROM sys.dm\_exec\_sessions WHERE is\_user\_process = 1;

GO

-- Step 4: Use that column to filter the currently executing requests

-- by joining the two tables on session\_id.

SELECT s.original\_login\_name, s.program\_name, r.command,

r.wait\_type, r.wait\_time, r.blocking\_session\_id, r.sql\_handle

FROM sys.dm\_exec\_requests AS r

INNER JOIN sys.dm\_exec\_sessions AS s

ON r.session\_id = s.session\_id

WHERE s.is\_user\_process = 1;

GO

-- Step 5: Note that we can also retrieve details of the SQL Batch that

-- is being executed, instead of just the handle. We do that by

-- using the sys.dm\_exec\_sql\_text function.

SELECT s.original\_login\_name, s.program\_name, r.command, t.text,

r.wait\_type, r.wait\_time, r.blocking\_session\_id

FROM sys.dm\_exec\_requests AS r

INNER JOIN sys.dm\_exec\_sessions AS s

ON r.session\_id = s.session\_id

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

WHERE s.is\_user\_process = 1;

GO

-- Step 6: Do not be too concerned about the complexity of the subquery

-- below but note that it is possible to find the actual statement

-- that is being executed rather than the batch.

SELECT s.original\_login\_name, s.program\_name, r.command,

(SELECT TOP (1) SUBSTRING(t.text, r.statement\_start\_offset / 2 + 1,

((CASE WHEN r.statement\_end\_offset = -1

THEN (LEN(CONVERT(nvarchar(max), t.text)) \* 2)

ELSE r.statement\_end\_offset

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

r.wait\_type, r.wait\_time, r.blocking\_session\_id

FROM sys.dm\_exec\_requests AS r

INNER JOIN sys.dm\_exec\_sessions AS s

ON r.session\_id = s.session\_id

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

WHERE s.is\_user\_process = 1;

GO

-- Step 7: Stop the script execution in the 91 - Workload.sql window

-- (Change to that window and click the Cancel Executing

-- Query toolbar button then return to this window)

-- Step 8: Investigate how the procedure cache is distributed at this point

SELECT cacheobjtype,

objtype ,

COUNT(\*) as CountofPlans,

SUM(usecounts) as UsageCount,

SUM(usecounts)/CAST(count(\*)as float) as AvgUsed ,

SUM(size\_in\_bytes)/1024./1024. as SizeinMB

FROM sys.dm\_exec\_cached\_plans

GROUP BY cacheobjtype, objtype

ORDER BY CountOfPlans DESC;

GO

-- Step 9: Locate the top 10 queries based on Average Reads

SELECT TOP (10) total\_logical\_reads/execution\_count AS AvgReads,

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

((CASE statement\_end\_offset

WHEN -1 THEN DATALENGTH(st.text)

ELSE qs.statement\_end\_offset END

- qs.statement\_start\_offset)/2) + 1) as StatementText

FROM sys.dm\_exec\_query\_stats AS qs

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

ORDER BY total\_logical\_reads/execution\_count DESC;

GO

-- Step 10: View I/O statistics for the database files

SELECT DB\_NAME(fs.database\_id) AS DatabaseName,

mf.name AS FileName,

mf.type\_desc,

fs.\*

FROM sys.dm\_io\_virtual\_file\_stats(NULL,NULL) AS fs

INNER JOIN sys.master\_files AS mf

ON fs.database\_id = mf.database\_id

AND fs.file\_id = mf.file\_id

ORDER BY fs.database\_id, fs.file\_id DESC;

GO

-- Step 11: View general wait statistics

SELECT \* FROM sys.dm\_os\_wait\_stats;

GO

select \*

from msdb.dbo.suspect\_pages

--size of db

exec sp\_spaceused

--size of db

SELECT DB\_NAME(database\_id) AS DatabaseName,

Name AS Logical\_Name,

Physical\_Name,

(size \* 8) / 1024 SizeMB

FROM sys.master\_files

WHERE DB\_NAME(database\_id) = 'ManufacturingDB'

Use DB2BMoved;

GO

Select \* from sys.database\_files;

|  |  |
| --- | --- |
| **Drive Letter and Directory** | **Purpose** |
| C: | System drive, Page file |
| L:\SQLLogs  K:\SQLLogs | SQL Server Log drive(s) |
| M:\SQLBackups  N:\SQLBackups | SQL Server Backup drive(s) |
| P:\SQLData  Q:\SQLData  R:\SQLData  S:SQLData | SQL Server Data drive(s) |
| T:\TempDB | SQL Server TempDB |
| Z: | Optical drive |

Table 1: Naming Standards and Structure for Logical Drives

<https://www.red-gate.com/simple-talk/sql/database-administration/provisioning-a-new-sql-server-instance-part-two/>

**Physical RAM**                    **MaxServerMem Setting**   
8GB                                        6000   
12GB                                     9000   
16GB                                     12000   
24GB                                     20000  
32GB                                     27000   
48GB                                     43000   
64GB                                     58000  
72GB                                     66000  
96GB                                     90000  
128GB                                   120000  
192GB                                   184000  
256GB                                   248000

--Database Backups

SELECT

CONVERT(CHAR(100), SERVERPROPERTY('Servername')) AS Server,

msdb.dbo.backupset.database\_name,

msdb.dbo.backupset.backup\_start\_date,

msdb.dbo.backupset.backup\_finish\_date,

msdb.dbo.backupset.expiration\_date,

CASE msdb..backupset.type

WHEN 'D' THEN 'Database'

WHEN 'L' THEN 'Log'

END AS backup\_type,

msdb.dbo.backupset.backup\_size,

msdb.dbo.backupmediafamily.logical\_device\_name,

msdb.dbo.backupmediafamily.physical\_device\_name,

msdb.dbo.backupset.name AS backupset\_name,

msdb.dbo.backupset.description

FROM msdb.dbo.backupmediafamily

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

WHERE (CONVERT(datetime, msdb.dbo.backupset.backup\_start\_date, 102) >= GETDATE() - 7)

and msdb.dbo.backupset.database\_name like '%natsoil%'

ORDER BY

msdb.dbo.backupset.database\_name,

msdb.dbo.backupset.backup\_finish\_date

--Database Backups

SELECT

BS.server\_name AS [Server Name]

,BS.database\_name AS [Database Name]

,BS.recovery\_model AS [Recovery Model]

,BMF.physical\_device\_name [Location]

,(CAST(BS.backup\_size / 1000000 AS INT)) AS [Size of Backup (MB)]

,CASE BS.[type] WHEN 'D' THEN 'Full'

WHEN 'I' THEN 'Differential'

WHEN 'L' THEN 'Transaction Log'

END AS [Type of Backup]

,BS.backup\_start\_date AS [Backup Date]

,BS.first\_lsn AS [First LSN]

,BS.last\_lsn AS [Last LSN]

FROM msdb.dbo.backupset BS

INNER JOIN msdb.dbo.backupmediafamily BMF ON BS.media\_set\_id = BMF.media\_set\_id

WHERE BS.database\_name like '%natsoil%'

ORDER BY backup\_start\_date DESC

,backup\_finish\_date