Contents

Find all database in instance and size

```
exec sp_databases
```

Find the Size of Database File and Log File

```
SELECT DB_NAME(database_id) AS DatabaseName,
Name AS Logical_Name,
Physical_Name, (size*8)/1024 SizeMB
FROM sys.master_files
ORDER BY 4 DESC
```

Last back of all databases

```
SELECT sdb.Name AS DatabaseName,
COALESCE(CONVERT(VARCHAR(12), MAX(bus.backup_finish_date), 101),'-') AS
LastBackUpTime
FROM sys.sysdatabases sdb
LEFT OUTER JOIN msdb.dbo.backupset bus ON bus.database_name = sdb.name
GROUP BY sdb.Name
```

List all users in Instance

```
select name,loginname from master.dbo.syslogins

Information about the current user in server level (login)
select system_user , suser_sname() , suser_sid()

Information about the current user in database level (db user)
select session_user , current_user , user , user_name() , user_id()
```

Database role and User

```
select rp.name as database_role, mp.name as database_user
from
sys.database_role_members drm
join
sys.database_principals rp on (drm.role_principal_id = rp.principal_id)
join
sys.database_principals mp on (drm.member_principal_id = mp.principal_id)
ORDER BY database_user
```

Find all users in Database

```
select * from
sys.database_principals
order by type_desc
```

To see when the Database was last accessed:

```
WITH agg AS
(
   SELECT
```

```
        max(last_user_seek) last_user_seek,
        max(last_user_scan) last_user_scan,
        max(last_user_lookup) last_user_lookup,
        max(last_user_update) last_user_update,
        sd.name dbname
   FROM
        sys.dm_db_index_usage_stats, master..sysdatabases sd
   WHERE
     database_id = sd.dbid  group by sd.name
)
SELECT
   dbname,
   last_read = MAX(last_read),
   last_write = MAX(last_write)
FROM
(
   SELECT dbname, last_user_seek, NULL FROM agg
   UNION ALL
   SELECT dbname, last_user_scan, NULL FROM agg
   UNION ALL
   SELECT dbname, last_user_lookup, NULL FROM agg
   UNION ALL
   SELECT dbname, NULL, last_user_update FROM agg
) AS x (dbname, last_read, last_write)
GROUP BY
   dbname
ORDER BY 2;
```

List of current users connected to the Server/Instance

```
SELECT login_name ,COUNT(session_id) AS session_count
FROM sys.dm_exec_sessions
GROUP BY login_name;
```

Last assesed database, read and write

```
WITH agg AS
(
   SELECT
        max(last_user_seek) last_user_seek,
        max(last_user_scan) last_user_scan,
        max(last_user_lookup) last_user_lookup,
        max(last_user_update) last_user_update,
        sd.name dbname
   FROM
        sys.dm_db_index_usage_stats, master..sysdatabases sd
   WHERE
     database_id = sd.dbid  group by sd.name
)
SELECT
   dbname,
   last_read = MAX(last_read),
   last_write = MAX(last_write)
FROM
```

```
(
   SELECT dbname, last_user_seek, NULL FROM agg
   UNION ALL
   SELECT dbname, last_user_scan, NULL FROM agg
   UNION ALL
   SELECT dbname, last_user_lookup, NULL FROM agg
   UNION ALL
   SELECT dbname, NULL, last_user_update FROM agg
) AS x (dbname, last_read, last_write)
GROUP BY
   dbname
ORDER BY 2;
```

Find Current Location of Data and Log File of All the Database

```
SELECT name, physical_name AS current_file_location
FROM sys.master_files
```

Find Space used by database (example ARSystem6)

```
USE ARSystem6;
GO
EXEC sp_spaceused @updateusage = N'TRUE';
GO
```

Recovery model of Databases in an instance

(Order by 2 – recovery model); (Order by 1- database name)

```
SELECT name AS 'Database Name',
recovery_model_desc AS 'Recovery Model'
FROM sys.databases
order by 2
GO
```

List of all last backup for all databases

```
CREATE PROCEDURE uspLastDatabaseBackupStats
AS
  SELECT   DatabaseName = b.database_name,
           LastBackupDate = a.backup_date,
           PhysicalDeviceName = physical_device_name,
           BackupSizeMB = convert(INT,backup_size),
           DurationMinutes = duration
  FROM     (SELECT   sd.name                    AS database_name,
                     MAX(bs.backup_finish_date) AS backup_date
            FROM     MASTER.dbo.sysdatabases sd

                     LEFT OUTER JOIN msdb.dbo.backupset bs
                       ON sd.name = bs.database_name
                     LEFT OUTER JOIN (SELECT   sd.name       AS database_name,
                     MAX(bs.backup_finish_date)   AS backup_date,
bm.physical_device_name, bs.backup_size / 1024 / 1024  AS backup_size,
DATEDIFF(mi,bs.backup_start_date, bs.backup_finish_date) AS duration
FROM     MASTER.dbo.sysdatabases sd
          LEFT OUTER JOIN msdb.dbo.backupset bs
                                  ON sd.name = bs.database_name
          LEFT OUTER JOIN msdb.dbo.backupmediafamily bm
                         ON bm.media_set_id = bs.media_set_id
GROUP BY    sd.name,
            bm.physical_device_name,
            bs.backup_size / 1024 / 1024,
DATEDIFF(mi,bs.backup_start_date,
            bs.backup_finish_date)) Summary
                       ON Summary.database_name = sd.name
                       AND Summary.backup_date = bs.backup_finish_date
            GROUP BY sd.name) a,
                     (SELECT   sd.name    AS database_name,
                     MAX(bs.backup_finish_date) AS backup_date,
                     Summary.physical_device_name,
                     Summary.backup_size,
                     Summary.duration
FROM     MASTER.dbo.sysdatabases sd
         LEFT OUTER JOIN msdb.dbo.backupset bs
         ON sd.name = bs.database_name
         LEFT OUTER JOIN (SELECT   sd.name         AS database_name,
         MAX(bs.backup_finish_date) AS backup_date,
            bm.physical_device_name,
            bs.backup_size / 1024 / 1024   AS backup_size,
DATEDIFF(mi,bs.backup_start_date,
         bs.backup_finish_date) AS duration
FROM     MASTER.dbo.sysdatabases sd
            LEFT OUTER JOIN msdb.dbo.backupset bs
            ON sd.name = bs.database_name
            LEFT OUTER JOIN msdb.dbo.backupmediafamily bm
            ON bm.media_set_id = bs.media_set_id
      GROUP BY sd.name,
             bm.physical_device_name,
             bs.backup_size / 1024 / 1024,
DATEDIFF(mi,bs.backup_start_date,
            bs.backup_finish_date)) Summary
            ON Summary.database_name = sd.name
            AND Summary.backup_date = bs.backup_finish_date
            GROUP BY sd.name,
                     bs.backup_finish_date,
                     Summary.physical_device_name,
                     Summary.backup_size,
                     Summary.duration) b

  WHERE    a.database_name = b.database_name
           AND a.backup_date = b.Backup_date
  ORDER BY DatabaseName
GO
EXEC uspLastDatabaseBackupStats
```

Fixed drive and space left

```
exec master..xp_fixeddrives
```

List of all Full Transactional and Differential Backup

```
declare @Server varchar(40)

set @server = CONVERT(varchar(35),
SERVERPROPERTY('machinename'))+'\'+isnull(CONVERT(varchar(35),
SERVERPROPERTY('instancename')),'DEFAULT')

--full backups
SELECT @server,
 fullrec.database_name,
 datediff(dd,fullrec.backup_finish_date,getdate()) as 'FullDays',
 fullrec.backup_finish_date  as 'FullFinish',
 datediff(dd,diffrec.backup_finish_date,getdate()) as 'DiffDays',
 diffrec.backup_finish_date  as 'DiffFinish',
 Case
    when diffrec.backup_finish_date is NULL then NULL
    else  datediff(dd,fullrec.backup_finish_date,diffrec.backup_finish_date)
 end as 'DaysBetwix',
 datediff(hh,tranrec.backup_finish_date,getdate()) as 'TranHours',
 tranrec.backup_finish_date  as 'TranFinish'

FROM msdb..backupset as fullrec

left outer join msdb..backupset as tranrec
on tranrec.database_name = fullrec.database_name
and tranrec.type = 'L'
and tranrec.backup_finish_date =
  ((select max(backup_finish_date)
      from msdb..backupset b2
     where b2.database_name = fullrec.database_name
       and b2.type = 'L'))

left outer join msdb..backupset as diffrec
on diffrec.database_name = fullrec.database_name
and diffrec.type = 'I'
and diffrec.backup_finish_date =
  ((select max(backup_finish_date)
      from msdb..backupset b2
    where b2.database_name = fullrec.database_name
       and b2.type = 'I'))

where fullrec.type = 'D' -- full backups only
and fullrec.backup_finish_date =
   (select max(backup_finish_date)
      from msdb..backupset b2
     where b2.database_name = fullrec.database_name
       and b2.type = 'D')


and fullrec.database_name in (select name from master..sysdatabases)
and fullrec.database_name not in ('tempdb','pubs','northwind','model')

-- never backed up
Union all
select @server
```

```
 ,name --,  '****  ','Red',
,9999,'1900-01-01 00:00:00',
NULL, NULL , NULL, NULL, NULL
from master..sysdatabases as record
where name not in (select distinct database_name from msdb..backupset)
and name not in ('tempdb','pubs','northwind','model')
order by 1,2
```

Grant execution permission on Function

```
GRANT EXECUTE ON Get_DownLoadCouncil_Func TO public
GRANT EXECUTE ON Get_DownLoadCouncil_Func TO user
```

Find User defined stored procedures

```
Select * from sys.objects where type='p' and is_ms_shipped=0 and [name] not like
'sp[_]%diagram%'
```

View stored procedures

```
sp_helptext spNewAvgGrade;
Explanation: In the above example, the sp_helptext displays the text of the
spNewAvgGrade stored procedure.

sp_help spNewAvgGrade;
Explanation: This example displays information about the stored procedure.

sp_depends spNewAvgGrade;
Explanation: This example shows the dependencies of the stored procedure.
--- P m R
```

Create Sp Job Stats

```
USE [??YourDatabaseOrMSDB??]
GO
/****** Object:  StoredProcedure [dbo].[sp_JobStats] ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[sp_JobStats] AS
EXEC msdb.dbo.sp_help_job @enabled=1
```

Truncate all tables

```
exec sys.sp_MSForEachtable 'TRUNCATE TABLE ?'
```

Create all database to simple

```
/*
    Only use this script for SQL Server development servers!
    Script must be executed as sysadmin

    This script will execute the following actions on all databases
        - set recovery model to [Simple]
        - truncate log file
        - shrink log file
*/

use [master]
go

-- Declare container variabels for each column we select in the cursor
declare @databaseName nvarchar(128)

-- Define the cursor name
declare databaseCursor cursor
-- Define the dataset to loop
for
select [name] from sys.databases

-- Start loop
open databaseCursor

-- Get information from the first row
fetch next from databaseCursor into @databaseName

-- Loop until there are no more rows
while @@fetch_status = 0
begin
    print 'Setting recovery model to Simple for database [' + @databaseName + ']'
    exec('alter database [' + @databaseName + '] set recovery Simple')

    print 'Shrinking logfile for database [' + @databaseName + ']'
    exec('
    use [' + @databaseName + '];' +'

    declare @logfileName nvarchar(128);
    set @logfileName = (
        select top 1 [name] from sys.database_files where [type] = 1
    );
    dbcc shrinkfile(@logfileName,1);
    ')

    -- Get information from next row
    fetch next from databaseCursor into @databaseName
end

-- End loop and clean up
close databaseCursor
deallocate databaseCursor
go
```

View active connection

```
-- By Application
SELECT
     CPU              = SUM(cpu_time)
    ,WaitTime         = SUM(total_scheduled_time)
    ,ElapsedTime      = SUM(total_elapsed_time)
    ,Reads            = SUM(num_reads)
    ,Writes           = SUM(num_writes)
    ,Connections      = COUNT(1)
    ,Program          = program_name
FROM sys.dm_exec_connections con
LEFT JOIN sys.dm_exec_sessions ses
    ON ses.session_id = con.session_id
GROUP BY program_name
ORDER BY cpu DESC

-- Group By User
SELECT
     CPU              = SUM(cpu_time)
    ,WaitTime         = SUM(total_scheduled_time)
    ,ElapsedTime      = SUM(total_elapsed_time)
    ,Reads            = SUM(num_reads)
    ,Writes           = SUM(num_writes)
    ,Connections      = COUNT(1)
    ,[login]          = original_login_name
from sys.dm_exec_connections con
LEFT JOIN sys.dm_exec_sessions ses
ON ses.session_id = con.session_id
GROUP BY original_login_name
```

Drop Orphan Users

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

ALTER Procedure [dbo].[DropOrphanUserAccounts]
as
BEGIN

set nocount on

create table #UserAccountsToDelete
(
ID int identity,
DB varchar(100),
[User] varchar(100)
)

DECLARE @DatabaseName nvarchar(100)
DECLARE my_DBs CURSOR FAST_FORWARD FOR
select [name] from master.sys.databases
WHERE [NAME] NOT IN ('MASTER','MODEL','TEMPDB','MSDB','ADVENTUREWORKS')
OPEN my_DBs
FETCH NEXT FROM my_DBs INTO @databasename
WHILE @@FETCH_STATUS = 0
Begin
```

```
insert #UserAccountsToDelete
exec('select '''+@databasename+''',sdp.name from
'+@DatabaseName+'.sys.database_principals sdp
left join sys.server_principals ssp on sdp.sid = ssp.sid
where ssp.sid is null and sdp.type in (''S'',''U'',''G'')
and sdp.name not in (''guest'', ''INFORMATION_SCHEMA'', ''sys'', ''BROKER_USER'',
''dbo'')')

FETCH NEXT FROM my_DBs INTO @databasename
END
CLOSE my_DBs
DEALLOCATE my_DBs

declare @contador int
declare @ciclo as int
declare @BD varchar(100)
declare @User varchar(100)
set @ciclo =1
set @contador =(select max(id) from #UserAccountsToDelete)

WHILE(@ciclo < = @contador )
begin

set @BD = (select db from #UserAccountsToDelete where id =@ciclo)
set @User = (select [user] from #UserAccountsToDelete where id =@ciclo)

begin try
exec ('use '+@bd+ ' drop schema ['+@user+']')
end try
begin catch
end catch
exec ('use '+@bd+ ' drop user ['+@user+']')

set @ciclo =@ciclo +1

end

drop table #UserAccountsToDelete
END
```

SQL Security Audit

```
USE master
GO
SET nocount ON

-- Get all roles
CREATE TABLE #temp_srvrole
(ServerRole VARCHAR(128), Description VARCHAR(128))
INSERT INTO #temp_srvrole
EXEC sp_helpsrvrole

-- sp_help syslogins
CREATE TABLE #temp_memberrole
(ServerRole VARCHAR(128),
MemberName VARCHAR(265),
MemberSID VARCHAR(300))

DECLARE @ServerRole VARCHAR(128)
DECLARE srv_role CURSOR FAST_FORWARD FOR
SELECT ServerRole FROM #temp_srvrole
OPEN srv_role
FETCH NEXT FROM srv_role INTO @ServerRole

WHILE @@FETCH_STATUS = 0
BEGIN
INSERT INTO #temp_memberrole
EXEC sp_helpsrvrolemember @ServerRole
FETCH NEXT FROM srv_role INTO @ServerRole
END

CLOSE srv_role
DEALLOCATE srv_role

SELECT ServerRole, MemberName FROM #temp_memberrole

-- IF BUILTIN\Administrators is exist and sysadmin
IF EXISTS(SELECT *FROM #temp_memberrole
WHERE MemberName = 'BUILTIN\Administrators'
AND ServerRole = 'sysadmin' )
BEGIN
CREATE TABLE #temp_localadmin (output VARCHAR(8000))
INSERT INTO #temp_localadmin
EXEC xp_cmdshell 'net localgroup administrators'

SELECT output AS local_administrator
FROM #temp_localadmin
WHERE output LIKE '%\%'
DROP TABLE #temp_localadmin
END

DROP TABLE #temp_srvrole
DROP TABLE #temp_memberrole

-- Get individual Logins
SELECT name, 'Individual NT Login' LoginType
FROM syslogins
WHERE isntgroup = 0 AND isntname = 1
UNION
SELECT name, 'Individual SQL Login' LoginType
FROM syslogins
WHERE isntgroup = 0 AND isntname = 0
UNION ALL
-- Get Group logins
SELECT name,'NT Group Login' LoginType
FROM syslogins
WHERE isntgroup = 1


-- get group list
-- EXEC xp_cmdshell 'net group "AnalyticsDev" /domain'
CREATE TABLE #temp_groupadmin
(output VARCHAR(8000))
CREATE TABLE #temp_groupadmin2
(groupName VARCHAR(256), groupMember VARCHAR(1000))
DECLARE @grpname VARCHAR(128)
DECLARE @sqlcmd VARCHAR(1000)

DECLARE grp_role CURSOR FAST_FORWARD FOR
SELECT REPLACE(name,'US\','')
FROM syslogins
WHERE isntgroup = 1 AND name LIKE 'US\%'
```

```
OPEN grp_role
FETCH NEXT FROM grp_role INTO @grpname

WHILE @@FETCH_STATUS = 0
BEGIN

SET @sqlcmd = 'net group "' + @grpname + '" /domain'
TRUNCATE TABLE #temp_groupadmin

PRINT @sqlcmd
INSERT INTO #temp_groupadmin
EXEC xp_cmdshell @sqlcmd

SET ROWCOUNT 8
DELETE FROM #temp_groupadmin

SET ROWCOUNT 0

INSERT INTO #temp_groupadmin2
SELECT @grpname, output FROM #temp_groupadmin
WHERE output NOT LIKE ('%The command completed successfully%')

FETCH NEXT FROM grp_role INTO @grpname
END

CLOSE grp_role
DEALLOCATE grp_role

SELECT * FROM #temp_groupadmin2

DROP TABLE #temp_groupadmin
DROP TABLE #temp_groupadmin2

PRINT 'EXEC sp_validatelogins '
PRINT '----------------------------------------------'
EXEC sp_validatelogins
PRINT ''

-- Get all the Database Rols for that specIFic members
CREATE TABLE #temp_rolemember
(DbRole VARCHAR(128),MemberName VARCHAR(128),MemberSID VARCHAR(1000))
CREATE TABLE #temp_rolemember_final
(DbName VARCHAR(100), DbRole VARCHAR(128),MemberName VARCHAR(128))

DECLARE @dbname VARCHAR(128)
DECLARE @sqlcmd2 VARCHAR(1000)

DECLARE grp_role CURSOR FOR
SELECT name FROM sysdatabases
WHERE name NOT IN ('tempdb')
AND DATABASEPROPERTYEX(name, 'Status') = 'ONLINE'

OPEN grp_role
FETCH NEXT FROM grp_role INTO @dbname

WHILE @@FETCH_STATUS = 0
BEGIN

TRUNCATE TABLE #temp_rolemember
SET @sqlcmd2 = 'EXEC [' + @dbname + ']..sp_helprolemember'

PRINT @sqlcmd2
INSERT INTO #temp_rolemember
EXECUTE(@sqlcmd2)
```

```
INSERT INTO #temp_rolemember_final
SELECT @dbname AS DbName, DbRole, MemberName
FROM #temp_rolemember

FETCH NEXT FROM grp_role INTO @dbname
END

CLOSE grp_role
DEALLOCATE grp_role

SELECT * FROM #temp_rolemember_final

DROP TABLE #temp_rolemember
DROP TABLE #temp_rolemember_final
```

Alter Username

```
alter user [nih\laxi] with name = [NIH\laxi]
```

Fix orphan Users

```
CREATE TABLE #OrphanedUsers(
row_num  INT IDENTITY(1,1),
username VARCHAR(1000),
id       VARCHAR(1000)
)
INSERT INTO #OrphanedUsers(username,id)
EXEC sp_change_users_login 'Report'

DECLARE @rowCount INT = (SELECT COUNT(1) FROM #OrphanedUsers );

DECLARE @i INT =1 ;
DECLARE @tempUsername VARCHAR(1000);

WHILE(@i <= @rowCount)
BEGIN
      SELECT @tempUsername = username FROM #OrphanedUsers WHERE row_num = @i;

      EXEC  sp_change_users_login 'Auto_Fix',@tempUsername;

      SET @i = @i+1;
END

DROP TABLE #OrphanedUsers;
```

Get step failure data stored procedure

```
USE [msdb]
GO
/****** Object:  StoredProcedure [dbo].[pr_GetStepFailureData]    Script Date:
09/21/2009 14:17:35 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[pr_GetStepFailureData]
(
@JobName VARCHAR(250)
)
AS
/*
This procedure gets failure log data for the failed step of a SQL Server Agent job
*/
DECLARE @job_id UNIQUEIDENTIFIER
SELECT @job_id = job_id FROM dbo.sysjobs WHERE [name] = @JobName
SELECT @JobName AS Job, 'Step ' + CAST(JH.step_id AS VARCHAR(3)) + ' of ' + (SELECT
CAST(COUNT(*) AS VARCHAR(5)) FROM dbo.sysjobsteps WHERE job_id = @job_id) AS
StepFailed,
 CAST(SUBSTRING(CAST(JH.run_date AS CHAR(8)),5,2) AS CHAR(2)) + '/' +
CAST(RIGHT(JH.run_date,2) AS CHAR(2)) + '/' + CAST(LEFT(JH.run_date,4) AS CHAR(4))
AS DateRun,
 LEFT(RIGHT('0' + CAST(JH.run_time AS VARCHAR(6)),6),2) + ':' + SUBSTRING(RIGHT('0'
+ CAST(JH.run_time AS VARCHAR(6)),6),3,2) + ':' + LEFT(RIGHT('0' + CAST(JH.run_time
AS VARCHAR(6)),6),2) AS TimeRun,
 JS.step_name,
 JH.run_duration,
 CASE
 WHEN JSL.[log] IS NULL THEN JH.[Message]
 ELSE JSL.[log]
 END AS LogOutput
FROM dbo.sysjobsteps JS INNER JOIN dbo.sysjobhistory JH
 ON JS.job_id = JH.job_id AND JS.step_id = JH.step_id
 LEFT OUTER JOIN dbo.sysjobstepslogs JSL
 ON JS.step_uid = JSL.step_uid
WHERE INSTANCE_ID >
 (SELECT MIN(INSTANCE_ID)
 FROM (
 SELECT top (2) INSTANCE_ID, job_id
 FROM dbo.sysjobhistory
 WHERE job_id = @job_id
 AND STEP_ID = 0
 ORDER BY INSTANCE_ID desc
 ) A
 )
 AND JS.step_id <> 0
 AND JH.job_id = @job_id
 AND JH.run_status = 0
ORDER BY JS.step_id ASC
```

Orphaned users for each database

```
/**************************************************
** Purpose: To return database users (for each db) orphaned from any login.
**************************************************/
--create a temp table to store the results
CREATE TABLE #temp (
DatabaseName NVARCHAR(50),
UserName NVARCHAR(50)
)
--create statement to run on each database
declare @sql nvarchar(500)
SET @sql='select ''?'' as DBName
, name AS UserName
from [?]..sysusers
where (sid is not null and sid <> 0x0)
and suser_sname(sid) is null and
(issqlrole <> 1) AND
(isapprole <> 1) AND
(name <> ''INFORMATION_SCHEMA'') AND
```

```
(name <> ''guest'') AND
(name <> ''sys'') AND
(name <> ''dbo'') AND
(name <> ''system_function_schema'')
order by name
'
--insert the results from each database to temp table
INSERT INTO #temp
exec SP_MSforeachDB @sql
--return results
SELECT * FROM #temp
Order by DatabaseName
DROP TABLE #temp
```

To find all servers including Linked server

```
select * from sys.servers

sp_linkedservers
```

to copy a stored procedure – from Management Studio

```
right click the stored procedure and copy to file. Save the file and run in the new
place where you want to store.

(names of database should be same)
```

Find the torn page. If no results returned then you are fine

```
SELECT db_name(database_id) DatabaseName, file_id, page_id, last_update_date
FROM msdb..suspect_pages
WHERE event_type = 3

--Backup your transaction log
USE master
BACKUP LOG DBName
TO DISK = 'C:\DBName.trn'
WITH NORECOVERY

--Restore Torn Page. 1 is the file_id, and 123 is the page_id from the first
query
RESTORE DATABASE DBName PAGE='1:123'
FROM DISK='C:\DBName.bak'
WITH NORECOVERY

--Restore your log
RESTORE LOG DBName FROM
DISK='C:\DBName.trn'
WITH RECOVERY
```

Create Procedure for LogCheckUp

```
Create Procedure usp_LogCheckUp
AS
SELECT db.[name] AS [Database Name] ,
db.recovery_model_desc AS [Recovery Model] ,
db.log_reuse_wait_desc AS [Log Reuse Wait Description] ,
ls.cntr_value AS [Log Size (KB)] ,
lu.cntr_value AS [Log Used (KB)] ,
CAST(CAST(lu.cntr_value AS FLOAT) / CAST(ls.cntr_value AS FLOAT)
AS DECIMAL(18,2)) * 100 AS [Log Used %] ,
db.[compatibility_level] AS [DB Compatibility Level] ,
db.page_verify_option_desc AS [Page Verify Option]
FROM sys.databases AS db
INNER JOIN sys.dm_os_performance_counters AS lu
ON db.name = lu.instance_name
INNER JOIN sys.dm_os_performance_counters AS ls
ON db.name = ls.instance_name
WHERE lu.counter_name LIKE 'Log File(s) Used Size (KB)%'
AND ls.counter_name LIKE 'Log File(s) Size (KB)%' ;
```

Restore and rebuilt Index

```
Use [Databasename}

-- Description : This script reorganizes and rebuilds the index if the
fragmentation level is higher the given threshold
-- You can define the threshold for reorganize as well as for rebuild and script
will work accordingly
-- INPUTS : @fillfactor - While rebuilding index what would be FILLFACTOR for new
index
-- @FragmentationThresholdForReorganizeTableLowerLimit - Fragmentation Level lower
threshold to check for reorganizing the table, if the fragmentation is higher than
this level, it will be considered for reorganize
-- @@FragmentationThresholdForRebuildTableLowerLimit - Fragmentation Level lower
threshold to check for rebuilding the table, if the fragmentation is higher than
this level, it will be considered for rebuild
-- NOTES : PRINT statements are all queued up and don't show up until the entire
script is printed. However, there is an alternative to PRINTing messages.
-- You can raise an error that isn't really an error (code of 0) and you'll get the
same effect--message will be printed immediately.
DECLARE @cmd NVARCHAR(1000)
DECLARE @Table VARCHAR(255)
DECLARE @SchemaName VARCHAR(255)
DECLARE @IndexName VARCHAR(255)
DECLARE @AvgFragmentationInPercent DECIMAL
DECLARE @fillfactor INT
```

```sql
DECLARE @FragmentationThresholdForReorganizeTableLowerLimit VARCHAR(10)
DECLARE @FragmentationThresholdForRebuildTableLowerLimit VARCHAR(10)
DECLARE @Message VARCHAR(1000)

SET NOCOUNT ON

--You can specify your customized value for reorganize and rebuild indexes, the
default values of 10 and 30 means index will be reorgnized if the fragmentation
level is more than equal to 10 and less than 30, if the fragmentation level is more
than equal to 30 then index will be rebuilt

SET @fillfactor = 90
SET @FragmentationThresholdForReorganizeTableLowerLimit = '10.0' -- Percent
SET @FragmentationThresholdForRebuildTableLowerLimit = '30.0' -- Percent

BEGIN TRY

-- ensure the temporary table does not exist
IF (SELECT OBJECT_ID('tempdb..#FramentedTableList')) IS NOT NULL
DROP TABLE #FramentedTableList;

SET @Message = 'DATE : ' + CONVERT(VARCHAR, GETDATE()) + ' - Retrieving indexes
with high fragmentation from ' + DB_NAME() + ' database.'
RAISERROR(@Message, 0, 1) WITH NOWAIT

SELECT OBJECT_NAME(IPS.OBJECT_ID) AS [TableName], avg_fragmentation_in_percent,
SI.name [IndexName],
schema_name(ST.schema_id) AS [SchemaName], 0 AS IsProcessed INTO
#FramentedTableList
FROM sys.dm_db_index_physical_stats(DB_ID(), NULL, NULL, NULL , NULL) IPS
JOIN sys.tables ST WITH (nolock) ON IPS.OBJECT_ID = ST.OBJECT_ID
JOIN sys.indexes SI WITH (nolock) ON IPS.OBJECT_ID = SI.OBJECT_ID AND IPS.index_id
= SI.index_id
WHERE ST.is_ms_shipped = 0 AND SI.name IS NOT NULL
AND avg_fragmentation_in_percent >= CONVERT(DECIMAL,
@FragmentationThresholdForReorganizeTableLowerLimit)
ORDER BY avg_fragmentation_in_percent DESC

SET @Message = 'DATE : ' + CONVERT(VARCHAR, GETDATE()) + ' - Retrieved indexes with
high fragmentation from ' + DB_NAME() + ' database.'

RAISERROR(@Message, 0, 1) WITH NOWAIT
RAISERROR('', 0, 1) WITH NOWAIT

WHILE EXISTS ( SELECT 1 FROM #FramentedTableList WHERE IsProcessed = 0 )
BEGIN

  SELECT TOP 1 @Table = TableName, @AvgFragmentationInPercent =
avg_fragmentation_in_percent,
  @SchemaName = SchemaName, @IndexName = IndexName
  FROM #FramentedTableList
  WHERE IsProcessed = 0

  --Reorganizing the index
  IF((@AvgFragmentationInPercent >=
@FragmentationThresholdForReorganizeTableLowerLimit) AND
(@AvgFragmentationInPercent < @FragmentationThresholdForRebuildTableLowerLimit))
```

```
  BEGIN
    SET @Message = 'DATE : ' + CONVERT(VARCHAR, GETDATE()) + ' - Reorganizing Index 
for [' + @Table + '] which has avg_fragmentation_in_percent = ' + CONVERT(VARCHAR, 
@AvgFragmentationInPercent) + '.'
    RAISERROR(@Message, 0, 1) WITH NOWAIT
    SET @cmd = 'ALTER INDEX ' + @IndexName + ' ON [' + RTRIM(LTRIM(@SchemaName)) + 
'].[' + RTRIM(LTRIM(@Table)) + '] REORGANIZE'
    EXEC (@cmd)
    --PRINT @cmd
    SET @Message = 'DATE : ' + CONVERT(VARCHAR, GETDATE()) + ' - Reorganize Index 
completed successfully for [' + @Table + '].'
    RAISERROR(@Message, 0, 1) WITH NOWAIT
    RAISERROR('', 0, 1) WITH NOWAIT
  END
  --Rebuilding the index
  ELSE IF (@AvgFragmentationInPercent >= 
@FragmentationThresholdForRebuildTableLowerLimit )
  BEGIN
    SET @Message = 'DATE : ' + CONVERT(VARCHAR, GETDATE()) + ' - Rebuilding Index 
for [' + @Table + '] which has avg_fragmentation_in_percent = ' + CONVERT(VARCHAR, 
@AvgFragmentationInPercent) + '.'
    RAISERROR(@Message, 0, 1) WITH NOWAIT
    SET @cmd = 'ALTER INDEX ' + @IndexName + ' ON [' + RTRIM(LTRIM(@SchemaName)) + 
'].[' + RTRIM(LTRIM(@Table)) + '] REBUILD WITH (FILLFACTOR = ' + 
CONVERT(VARCHAR(3),@fillfactor) + ', STATISTICS_NORECOMPUTE = OFF)'
    EXEC (@cmd)
    --PRINT @cmd
    SET @Message = 'DATE : ' + CONVERT(VARCHAR, GETDATE()) + ' - Rebuild Index 
completed successfully for [' + @Table + '].'
    RAISERROR(@Message, 0, 1) WITH NOWAIT
    RAISERROR('', 0, 1) WITH NOWAIT
  END

  UPDATE #FramentedTableList
  SET IsProcessed = 1
  WHERE TableName = @Table
  AND IndexName = @IndexName
END

DROP TABLE #FramentedTableList

END TRY

BEGIN CATCH
  PRINT 'DATE : ' + CONVERT(VARCHAR, GETDATE()) + ' There is some run time 
exception.'
  PRINT 'ERROR CODE : ' + CONVERT(VARCHAR, ERROR_NUMBER())
  PRINT 'ERROR MESSAGE : ' + ERROR_MESSAGE()
END CATCH
```

Disk Space Alert per diskdrive (stored procedure)

```
USE [master]
GO
```

```
/****** Object:  StoredProcedure [dbo].[usp_DiskSpaceAlert]    Script Date:
10/03/2011 15:50:05 ******/
IF  EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[usp_DiskSpaceAlert]') AND type in (N'P', N'PC'))
DROP PROCEDURE [dbo].[usp_DiskSpaceAlert]
GO

USE [master]
GO

/****** Object:  StoredProcedure [dbo].[usp_DiskSpaceAlert]    Script Date:
10/03/2011 15:50:05 ******/
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[usp_DiskSpaceAlert] @MinMBFree int, @Drive char(1), @RCPT
VARCHAR(500) AS

/*****************************************************
*      Object:  dbo.usp_DiskSpaceAlert (DiskSpaceAlert per diskdrive)
*      Dependent Objects: master.sys.xp_fixeddrives
*      Version: 1.0
*      Purpose: Validate sufficient disk space per drive
*      Detailed Description: Validate sufficient disk space based on based on the
@MinMBFree and @Drive parameters
*            Mails when defined amount is reached to parameter @RCPT
*      EXECUTE AS:
*            EXEC dbo.usp_DiskSpaceAlert @MinMBFree= 30000, @Drive='C',
@RCPT='someone@mail.com'
*      Updates:
*       v1.0 - Drive will be checked and sends mail when diskspace is less then
given amount
*****************************************************/

SET NOCOUNT ON
-- 1 - Declare variables
DECLARE @MBfree int
-- 2 - Initialize variables
SET @MBfree = 0
-- 3 - Create temp tables
CREATE TABLE #tbl_xp_fixeddrives
(Drive varchar(2) NOT NULL,
[MB free] int NOT NULL)
-- 4 - Populate #tbl_xp_fixeddrives
INSERT INTO #tbl_xp_fixeddrives(Drive, [MB free])
EXEC master.sys.xp_fixeddrives
-- 5 - Initialize the @MBfree value
SELECT @MBfree = [MB free]
FROM #tbl_xp_fixeddrives
WHERE Drive = @Drive
-- 6 - Determine if sufficient free space is available
IF @MBfree > @MinMBFree
 BEGIN
  RETURN
 END
ELSE
 BEGIN
  IF CHARINDEX('@',@RCPT) > 0  --THERE IS AN @ SYMBOL IN THE RECIPIENT - SEND EMAIL
      BEGIN
            DECLARE @MSG VARCHAR(400)
            SET @MSG =  @Drive + ' drive has only ' + CONVERT(VARCHAR,@MBfree) --
PUT THE VARS INTO A MSG
                  + 'MB (' +CONVERT(VARCHAR,@MBfree/1024)+ 'GB) left on ' +
@@SERVERNAME + CHAR(13) + CHAR(10)
            DECLARE @EMAIL VARCHAR(800)
            SET @EMAIL = 'EXEC msdb.dbo.sp_send_dbmail
                  @recipients = ''' + @RCPT + ''',
                  @body = ''' + @MSG + ''',
                  @subject = ''!! LOW FREE DISK SPACE ON DRIVE ' + @Drive + ' @ ' +
@@SERVERNAME + ' !!'''
            EXEC (@EMAIL)
      END
 END
-- 7 - DROP TABLE #tbl_xp_fixeddrives
DROP TABLE #tbl_xp_fixeddrives
SET NOCOUNT OFF
GO
```

Find location and time of last backup

```
SELECT
physical_device_name,
backup_start_date,
backup_finish_date,
backup_size/1024.0
AS BackupSizeKB
FROM msdb.dbo.backupset b
JOIN msdb.dbo.backupmediafamily m ON b.media_set_id = m.media_set_id
WHERE database_name = 'TTS'
ORDER BY backup_finish_date DESC
```

Find table sizes in a database

```
use TTS -here database name
go
create table #tablesize
(name nvarchar(120),rows char(11),reserved varchar(18),
data varchar(18),index_size varchar(18),unused varchar(18))
go
insert into #tablesize
exec
sp_msforeachtable @command1=
'exec sp_spaceused ''?'''
select * from #tablesize
go
drop table #tablesize
go
```

To Kill all processes

```
USE master;
GO
ALTER DATABASE TTS --database name here
SET SINGLE_USER
WITH ROLLBACK IMMEDIATE;
ALTER DATABASE TTS --database name here
SET MULTI_USER;
GO
```

SQL Server Users that are connected to your Database

```
SELECT
        login_name, count(session_id) as session_count
    FROM  sys.dm_exec_sessions
    GROUP BY login_name
```

Diagnostic Information Queries

Instance level queries

SQL and OS Version information for current instance

```
SELECT @@VERSION AS [SQL Server and OS Version Info];
-- SQL Server 2008 RTM is considered an "unsupported service pack" as of April 13,
2010
```

Windows information (SQL Server 2008 R2 SP1 or greater)

```
SELECT windows_release, windows_service_pack_level,
       windows_sku, os_language_version
FROM sys.dm_os_windows_info OPTION (RECOMPILE);
-- Gives you major OS version, Service Pack, Edition, and language info for the
operating system
```

SQL Server Services information (SQL Server 2008 R2 SP1 or greater)

```
SELECT servicename, startup_type_desc, status_desc,
last_startup_time, service_account, is_clustered, cluster_nodename
FROM sys.dm_server_services OPTION (RECOMPILE);
-- Tells you the account being used for the SQL Server Service and the SQL Agent
Service
-- Shows when they were last started, and their current status
-- Shows whether you are running on a failover cluster
```

When was SQL Server installed

```
SELECT createdate AS [SQL Server Install Date]
FROM sys.syslogins
WHERE [sid] = 0x010100000000000512000000;
-- Tells you the date and time that SQL Server was installed
-- I think it is a good idea to know how old your instance is
--- [sid] = 0x010100000000000512000000 sid of the login 'NT AUTHORITY\SYSTEM' Which get
created when you install SQL server
Or this will work best if u have restored master database after installing SQL
server.

SELECT createdate AS [SQL Server Install Date]
FROM sys.syslogins
WHERE name = 'NT AUTHORITY\SYSTEM' -----the login which gets created when you install SQL
Server.
```

Hardware information from SQL Server 2008 and 2008 R2

```
-- (Cannot distinguish between HT and multi-core)
SELECT cpu_count AS [Logical CPU Count], hyperthread_ratio AS [Hyperthread Ratio],
cpu_count/hyperthread_ratio AS [Physical CPU Count],
physical_memory_in_bytes/1048576 AS [Physical Memory (MB)],
sqlserver_start_time --, affinity_type_desc -- (affinity_type_desc is only in 2008
R2)
FROM sys.dm_os_sys_info OPTION (RECOMPILE);
-- Gives you some good basic hardware information about your database server
```

Get System Manufacturer and model number from

```
-- SQL Server Error log. This query might take a few seconds
-- if you have not recycled your error log recently
EXEC xp_readerrorlog 0,1,"Manufacturer";
-- This can help you determine the capabilities
-- and capacities of your database server
```

Get processor description from Windows Registry

```
EXEC xp_instance_regread
'HKEY_LOCAL_MACHINE',
'HARDWARE\DESCRIPTION\System\CentralProcessor\0',
'ProcessorNameString';
-- Gives you the model number and rated clock speed of your processor(s)
```

Get configuration values for instance

```
SELECT name, value, value_in_use, [description]
FROM sys.configurations
ORDER BY name OPTION (RECOMPILE);
-- Focus on
-- backup compression default
-- clr enabled (only enable if it is needed)
-- lightweight pooling (should be zero)
-- max degree of parallelism (depends on your workload)
-- max server memory (MB) (set to an appropriate value)
-- optimize for ad hoc workloads (should be 1)
-- priority boost (should be zero)
```

File Names and Paths for TempDB and all user databases in instance

```
SELECT DB_NAME([database_id])AS [Database Name],
       [file_id], name, physical_name, type_desc, state_desc,
       CONVERT( bigint, size/128.0) AS [Total Size in MB]
FROM sys.master_files
WHERE [database_id] > 4
AND [database_id] <> 32767
OR [database_id] = 2
ORDER BY DB_NAME([database_id]) OPTION (RECOMPILE);
-- Things to look at:
-- Are data files and log files on different drives?
-- Is everything on the C: drive?
-- Is TempDB on dedicated drives?
-- Is there only one TempDB data file?
-- Are all of the TempDB data files the same size?
-- Are there multiple data files for user databases?
```

Volume info for all databases on the current instance

```
SELECT DB_NAME(f.database_id) AS [DatabaseName], f.file_id,
vs.volume_mount_point, vs.total_bytes, vs.available_bytes,
CAST(CAST(vs.available_bytes AS FLOAT)/ CAST(vs.total_bytes AS FLOAT) AS
DECIMAL(18,1)) * 100 AS [Space Free %]
FROM sys.master_files AS f
CROSS APPLY sys.dm_os_volume_stats(f.database_id, f.file_id) AS vs
ORDER BY f.database_id OPTION (RECOMPILE);
--Shows you the free space on the LUNs where you have database data or log files
```

Recovery model, log reuse wait description, log file size, log usage size and compatibility level for all databases on instance

```
SELECT db.[name] AS [Database Name], db.recovery_model_desc AS [Recovery Model],
db.log_reuse_wait_desc AS [Log Reuse Wait Description],
ls.cntr_value AS [Log Size (KB)], lu.cntr_value AS [Log Used (KB)],
CAST(CAST(lu.cntr_value AS FLOAT) / CAST(ls.cntr_value AS FLOAT)AS DECIMAL(18,2)) *
100 AS [Log Used %],
db.[compatibility_level] AS [DB Compatibility Level],
db.page_verify_option_desc AS [Page Verify Option], db.is_auto_create_stats_on,
db.is_auto_update_stats_on,
db.is_auto_update_stats_async_on, db.is_parameterization_forced,
db.snapshot_isolation_state_desc, db.is_read_committed_snapshot_on,
db.is_auto_close_on, db.is_auto_shrink_on
FROM sys.databases AS db
INNER JOIN sys.dm_os_performance_counters AS lu
ON db.name = lu.instance_name
INNER JOIN sys.dm_os_performance_counters AS ls
ON db.name = ls.instance_name
WHERE lu.counter_name LIKE N'Log File(s) Used Size (KB)%'
AND ls.counter_name LIKE N'Log File(s) Size (KB)%'
AND ls.cntr_value > 0 OPTION (RECOMPILE);
-- Things to look at:
-- How many databases are on the instance?
-- What recovery models are they using?
-- What is the log reuse wait description?
-- How full are the transaction logs ?
-- What compatibility level are they on?
-- What is the Page Verify Option?
-- Make sure auto_shrink and auto_close are not enabled!
```

Calculates average stalls per read, per write, and per total input/output for each database file.

```
SELECT DB_NAME(fs.database_id) AS [Database Name], mf.physical_name,
io_stall_read_ms, num_of_reads,
CAST(io_stall_read_ms/(1.0 + num_of_reads) AS NUMERIC(10,1)) AS
[avg_read_stall_ms],io_stall_write_ms,
num_of_writes,CAST(io_stall_write_ms/(1.0+num_of_writes) AS NUMERIC(10,1)) AS
[avg_write_stall_ms],
io_stall_read_ms + io_stall_write_ms AS [io_stalls], num_of_reads + num_of_writes
AS [total_io],
CAST((io_stall_read_ms + io_stall_write_ms)/(1.0 + num_of_reads + num_of_writes) AS
NUMERIC(10,1))
AS [avg_io_stall_ms]
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 avg_io_stall_ms DESC OPTION (RECOMPILE);
-- Helps you determine which database files on the entire instance have the most
I/O bottlenecks
```

Get CPU utilization by database (adapted from Robert Pearl)

```
WITH DB_CPU_Stats
AS
(SELECT DatabaseID, DB_Name(DatabaseID) AS [DatabaseName], SUM(total_worker_time)
AS [CPU_Time_Ms]
 FROM sys.dm_exec_query_stats AS qs
 CROSS APPLY (SELECT CONVERT(int, value) AS [DatabaseID]
              FROM sys.dm_exec_plan_attributes(qs.plan_handle)
              WHERE attribute = N'dbid') AS F_DB
 GROUP BY DatabaseID)
SELECT ROW_NUMBER() OVER(ORDER BY [CPU_Time_Ms] DESC) AS [row_num],
       DatabaseName, [CPU_Time_Ms],
       CAST([CPU_Time_Ms] * 1.0 / SUM([CPU_Time_Ms]) OVER() * 100.0 AS DECIMAL(5,
2)) AS [CPUPercent]
FROM DB_CPU_Stats
WHERE DatabaseID > 4 -- system databases
AND DatabaseID <> 32767 -- ResourceDB
ORDER BY row_num OPTION (RECOMPILE);
-- Helps determine which database is using the most CPU resources on the instance
```

Get total buffer usage by database for current instance

```
-- This make take some time to run on a busy instance
SELECT DB_NAME(database_id) AS [Database Name],
COUNT(*) * 8/1024.0 AS [Cached Size (MB)]
```

```
FROM sys.dm_os_buffer_descriptors
WHERE database_id > 4 -- system databases
AND database_id <> 32767 -- ResourceDB
GROUP BY DB_NAME(database_id)
ORDER BY [Cached Size (MB)] DESC OPTION (RECOMPILE);
-- Tells you how much memory (in the buffer pool) is being used by each database on
the instance
-- Clear Wait Stats
-- DBCC SQLPERF('sys.dm_os_wait_stats', CLEAR);
```

Isolate top waits for server instance since last restart or statistics clear

```
WITH Waits AS
(SELECT wait_type, wait_time_ms / 1000. AS wait_time_s,
100. * wait_time_ms / SUM(wait_time_ms) OVER() AS pct,
ROW_NUMBER() OVER(ORDER BY wait_time_ms DESC) AS rn
FROM sys.dm_os_wait_stats
WHERE wait_type NOT IN
('CLR_SEMAPHORE','LAZYWRITER_SLEEP','RESOURCE_QUEUE','SLEEP_TASK',
'SLEEP_SYSTEMTASK','SQLTRACE_BUFFER_FLUSH','WAITFOR',
'LOGMGR_QUEUE','CHECKPOINT_QUEUE',
'REQUEST_FOR_DEADLOCK_SEARCH','XE_TIMER_EVENT','BROKER_TO_FLUSH','BROKER_TASK_STOP'
,'CLR_MANUAL_EVENT',
'CLR_AUTO_EVENT','DISPATCHER_QUEUE_SEMAPHORE', 'FT_IFTS_SCHEDULER_IDLE_WAIT',
'XE_DISPATCHER_WAIT', 'XE_DISPATCHER_JOIN', 'SQLTRACE_INCREMENTAL_FLUSH_SLEEP',
'ONDEMAND_TASK_QUEUE', 'BROKER_EVENTHANDLER', 'SLEEP_BPOOL_FLUSH'))
SELECT W1.wait_type,
CAST(W1.wait_time_s AS DECIMAL(12, 2)) AS wait_time_s,
CAST(W1.pct AS DECIMAL(12, 2)) AS pct,
CAST(SUM(W2.pct) AS DECIMAL(12, 2)) AS running_pct
FROM Waits AS W1
INNER JOIN Waits AS W2
ON W2.rn <= W1.rn
GROUP BY W1.rn, W1.wait_type, W1.wait_time_s, W1.pct
HAVING SUM(W2.pct) - W1.pct < 99 OPTION (RECOMPILE); -- percentage threshold
-- Common Significant Wait types with BOL explanations

-- *** Network Related Waits ***
-- ASYNC_NETWORK_IO    Occurs on network writes when the task is blocked behind the
network

-- *** Locking Waits ***
-- LCK_M_IX    Occurs when a task is waiting to acquire an Intent Exclusive (IX)
lock
-- LCK_M_IU    Occurs when a task is waiting to acquire an Intent Update (IU) lock
-- LCK_M_S     Occurs when a task is waiting to acquire a Shared lock

-- *** I/O Related Waits ***
-- ASYNC_IO_COMPLETION Occurs when a task is waiting for I/Os to -----finish
-- IO_COMPLETION Occurs while waiting for I/O operations to complete.
-- This wait type generally represents non-data page I/Os.
-- Data page I/O completion waits appear as PAGEIOLATCH_* waits
-- PAGEIOLATCH_SH Occurs when a task is waiting on a latch for a buffer that is in
an I/O request.
-- The latch request is in Shared mode. Long waits may indicate problems with the
disk subsystem.
```

```
-- PAGEIOLATCH_EX Occurs when a task is waiting on a latch for a buffer that is in
an I/O request.
-- The latch request is in Exclusive mode. Long waits may indicate problems with
the disk subsystem.
-- WRITELOG Occurs while waiting for a log flush to complete.
-- Common operations that cause log flushes are checkpoints and transaction
commits.
-- PAGELATCH_EX Occurs when a task is waiting on a latch for a buffer that is not
in an I/O request.
-- The latch request is in Exclusive mode.
-- BACKUPIO Occurs when a backup task is waiting for data, or is waiting for a
buffer in which to store data

-- *** CPU Related Waits ***
-- SOS_SCHEDULER_YIELD  Occurs when a task voluntarily yields the scheduler for
other tasks to execute.
--   During this wait the task is waiting for its quantum to be renewed.

-- THREADPOOL   Occurs when a task is waiting for a worker to run on.
-- This can indicate that the maximum worker setting is too low, or that batch
executions are taking unusually long, thus reducing the number of workers available
to satisfy other batches.
-- CX_PACKET Occurs when trying to synchronize the query processor exchange
iterator
-- You may consider lowering the degree of parallelism if contention on this wait
type becomes a problem
--Often caused by missing indexes or poorly written queries
```

Signal Waits for instance

```
SELECT CAST(100.0 * SUM(signal_wait_time_ms) / SUM (wait_time_ms) AS NUMERIC(20,2))
AS [%signal (cpu) waits],
CAST(100.0 * SUM(wait_time_ms - signal_wait_time_ms) / SUM (wait_time_ms) AS
NUMERIC(20,2))
AS [%resource waits]
FROM sys.dm_os_wait_stats WITH (NOLOCK) OPTION (RECOMPILE);
-- Signal Waits above 10-15% is usually a sign of CPU pressure
```

Get logins that are connected and how many sessions they have

```
SELECT login_name, COUNT(session_id) AS [session_count]
FROM sys.dm_exec_sessions WITH (NOLOCK)
WHERE session_id > 50  -- filter out system SPIDs
GROUP BY login_name
ORDER BY COUNT(session_id) DESC OPTION (RECOMPILE);
-- This can help characterize your workload and
-- determine whether you are seeing a normal level of activity
```

Get Average Task Counts (run multiple times)

```
SELECT AVG(current_tasks_count) AS [Avg Task Count],
AVG(runnable_tasks_count) AS [Avg Runnable Task Count],
AVG(pending_disk_io_count) AS [AvgPendingDiskIOCount]
FROM sys.dm_os_schedulers WITH (NOLOCK)
WHERE scheduler_id < 255 OPTION (RECOMPILE);
```

```
-- Sustained values above 10 suggest further investigation in that area
-- High current_tasks_count is often an indication of locking/blocking problems
-- High runnable_tasks_count is an indication of CPU pressure
-- High pending_disk_io_count is an indication of I/O pressure
```

Get CPU Utilization History for last 256 minutes (in one minute intervals)

```
-- This version works with SQL Server 2008 and SQL Server 2008 R2 only
DECLARE @ts_now bigint = (SELECT cpu_ticks/(cpu_ticks/ms_ticks)FROM
sys.dm_os_sys_info);

SELECT TOP(256) SQLProcessUtilization AS [SQL Server Process CPU Utilization],
               SystemIdle AS [System Idle Process],
               100 - SystemIdle - SQLProcessUtilization AS [Other Process CPU
Utilization],
               DATEADD(ms, -1 * (@ts_now - [timestamp]), GETDATE()) AS [Event Time]
FROM (
          SELECT record.value('(./Record/@id)[1]', 'int') AS record_id,

        record.value('(./Record/SchedulerMonitorEvent/SystemHealth/SystemIdle)[1]',
'int')
                       AS [SystemIdle],

        record.value('(./Record/SchedulerMonitorEvent/SystemHealth/ProcessUtilizatio
n)[1]',
                       'int')
                       AS [SQLProcessUtilization], [timestamp]
          FROM (
                       SELECT [timestamp], CONVERT(xml, record) AS [record]
                       FROM sys.dm_os_ring_buffers WITH (NOLOCK)
                       WHERE ring_buffer_type = N'RING_BUFFER_SCHEDULER_MONITOR'
                       AND record LIKE N'%<SystemHealth>%') AS x
          ) AS y
ORDER BY record_id DESC OPTION (RECOMPILE);
-- Look at the trend over the entire period.
-- Also look at high sustained Other Process CPU Utilization values
```

Good basic information about memory amounts and state

```
SELECT total_physical_memory_kb, available_physical_memory_kb,
       total_page_file_kb, available_page_file_kb,
       system_memory_state_desc
FROM sys.dm_os_sys_memory WITH (NOLOCK) OPTION (RECOMPILE);
-- You want to see "Available physical memory is high"
-- This indicates that you are not under external memory pressure
```

SQL Server Process Address space info

```
--(shows whether locked pages is enabled, among other things)
SELECT physical_memory_in_use_kb,locked_page_allocations_kb,
       page_fault_count, memory_utilization_percentage,
       available_commit_limit_kb, process_physical_memory_low,
       process_virtual_memory_low
FROM sys.dm_os_process_memory WITH (NOLOCK) OPTION (RECOMPILE);
```

```
-- You want to see 0 for process_physical_memory_low
-- You want to see 0 for process_virtual_memory_low
-- This indicates that you are not under internal memory pressure
```

Page Life Expectancy (PLE) value for current instance

```
SELECT @@SERVERNAME AS [Server Name], [object_name], cntr_value AS [Page Life
Expectancy]
FROM sys.dm_os_performance_counters WITH (NOLOCK)
WHERE [object_name] LIKE N'%Buffer Manager%' -- Handles named instances
AND counter_name = N'Page life expectancy' OPTION (RECOMPILE);
-- PLE is a good measurement of memory pressure.
-- Higher PLE is better. Watch the trend, not the absolute value.
```

Memory Grants Outstanding value for current instance

```
SELECT @@SERVERNAME AS [Server Name], [object_name], cntr_value AS [Memory Grants
Outstanding]
FROM sys.dm_os_performance_counters WITH (NOLOCK)
WHERE [object_name] LIKE N'%Memory Manager%' -- Handles named instances
AND counter_name = N'Memory Grants Outstanding' OPTION (RECOMPILE);
-- Memory Grants Outstanding above zero for a sustained period is a very strong
indicator of memory pressure
```

Memory Grants Pending value for current instance

```
SELECT @@SERVERNAME AS [Server Name], [object_name], cntr_value AS [Memory Grants
Pending]
FROM sys.dm_os_performance_counters WITH (NOLOCK)
WHERE [object_name] LIKE N'%Memory Manager%' -- Handles named instances
AND counter_name = N'Memory Grants Pending' OPTION (RECOMPILE);
-- Memory Grants Pending above zero for a sustained period is a very strong
indicator of memory pressure
```

Memory Clerk Usage for instance

```
-- Look for high value for CACHESTORE_SQLCP (Ad-hoc query plans)
SELECT TOP(10) [type] AS [Memory Clerk Type], SUM(single_pages_kb) AS [SPA Mem, Kb]
FROM sys.dm_os_memory_clerks WITH (NOLOCK)
GROUP BY [type]
ORDER BY SUM(single_pages_kb) DESC OPTION (RECOMPILE);
-- CACHESTORE_SQLCP SQL Plans
-- These are cached SQL statements or batches that
-- aren't in stored procedures, functions and triggers
-- CACHESTORE_OBJCP Object Plans
-- These are compiled plans for
-- stored procedures, functions and triggers
-- CACHESTORE_PHDR Algebrizer Trees
-- An algebrizer tree is the parsed SQL text
-- that resolves the table and column names
```

Find single-use, ad-hoc queries that are bloating the plan cache

```
SELECT TOP(50) [text] AS [QueryText], cp.size_in_bytes
```

```
FROM sys.dm_exec_cached_plans AS cp WITH (NOLOCK)
CROSS APPLY sys.dm_exec_sql_text(plan_handle)
WHERE cp.cacheobjtype = N'Compiled Plan'
AND cp.objtype = N'Adhoc'
AND cp.usecounts = 1
ORDER BY cp.size_in_bytes DESC OPTION (RECOMPILE);
-- Gives you the text and size of single-use ad-hoc queries  that waste space in
the plan cache
-- Enabling 'optimize for ad hoc workloads' for the instance can help (SQL Server
2008 and 2008 R2 only)
-- Enabling forced parameterization for the database can help, but test first!
```

Database specific queries

Switch to a user database

```
USE YourDatabaseName;
GO
```

Individual File Sizes and space available for current database

```
SELECT name AS [File Name], [file_id], physical_name AS [Physical 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 WITH (NOLOCK) OPTION (RECOMPILE);
-- Look at how large and how full the files are and where they are located
-- Make sure the transaction log is not full!!
```

I/O Statistics by file for the current database

```
SELECT DB_NAME(DB_ID()) AS [Database Name],[file_id], num_of_reads, num_of_writes,
io_stall_read_ms, io_stall_write_ms,
CAST(100. * io_stall_read_ms/(io_stall_read_ms + io_stall_write_ms) AS
DECIMAL(10,1)) AS [IO Stall Reads Pct],
CAST(100. * io_stall_write_ms/(io_stall_write_ms + io_stall_read_ms) AS
DECIMAL(10,1)) AS [IO Stall Writes Pct],
(num_of_reads + num_of_writes) AS [Writes + Reads], num_of_bytes_read,
num_of_bytes_written,
CAST(100. * num_of_reads/(num_of_reads + num_of_writes) AS DECIMAL(10,1)) AS [#
Reads Pct],
CAST(100. * num_of_writes/(num_of_reads + num_of_writes) AS DECIMAL(10,1)) AS [#
Write Pct],
CAST(100. * num_of_bytes_read/(num_of_bytes_read + num_of_bytes_written) AS
DECIMAL(10,1)) AS [Read Bytes Pct],
CAST(100. * num_of_bytes_written/(num_of_bytes_read + num_of_bytes_written) AS
DECIMAL(10,1)) AS [Written Bytes Pct]
FROM sys.dm_io_virtual_file_stats(DB_ID(), NULL) OPTION (RECOMPILE);
-- This helps you characterize your workload better from an I/O perspective
-- Get VLF count for transaction log for the current database,
-- number of rows equals VLF count. Lower is better!
DBCC LOGINFO;
-- High VLF counts can affect write performance and they can make database restore
and recovery take much longer
```

Top Cached SPs By Execution Count (SQL 2008)

```
SELECT TOP(250) p.name AS [SP Name], qs.execution_count,
ISNULL(qs.execution_count/DATEDIFF(Second, qs.cached_time, GETDATE()), 0) AS
[Calls/Second],
qs.total_worker_time/qs.execution_count AS [AvgWorkerTime], qs.total_worker_time AS
[TotalWorkerTime],
qs.total_elapsed_time, qs.total_elapsed_time/qs.execution_count AS
[avg_elapsed_time],
qs.cached_time
FROM sys.procedures AS p WITH (NOLOCK)
INNER JOIN sys.dm_exec_procedure_stats AS qs WITH (NOLOCK)
ON p.[object_id] = qs.[object_id]
WHERE qs.database_id = DB_ID()
ORDER BY qs.execution_count DESC OPTION (RECOMPILE);
-- Tells you which cached stored procedures are called the most often
-- This helps you characterize and baseline your workload
```

Top Cached SPs By Avg Elapsed Time (SQL 2008)

```
SELECT TOP(25) p.name AS [SP Name], qs.total_elapsed_time/qs.execution_count AS
[avg_elapsed_time],
qs.total_elapsed_time, qs.execution_count,
ISNULL(qs.execution_count/DATEDIFF(Second, qs.cached_time,
GETDATE()), 0) AS [Calls/Second], qs.total_worker_time/qs.execution_count AS
[AvgWorkerTime],
qs.total_worker_time AS [TotalWorkerTime], qs.cached_time
FROM sys.procedures AS p WITH (NOLOCK)
INNER JOIN sys.dm_exec_procedure_stats AS qs WITH (NOLOCK)
ON p.[object_id] = qs.[object_id]
WHERE qs.database_id = DB_ID()
ORDER BY avg_elapsed_time DESC OPTION (RECOMPILE);
-- This helps you find long-running cached stored procedures that
-- may be easy to optimize with standard query tuning techniques
```

Top Cached SPs By Avg Elapsed Time with execution time variability (SQL 2008)

```
SELECT TOP(25) p.name AS [SP Name], qs.execution_count, qs.min_elapsed_time,
qs.total_elapsed_time/qs.execution_count AS [avg_elapsed_time],
qs.max_elapsed_time, qs.last_elapsed_time,  qs.cached_time
FROM sys.procedures AS p WITH (NOLOCK)
INNER JOIN sys.dm_exec_procedure_stats AS qs WITH (NOLOCK)
ON p.[object_id] = qs.[object_id]
WHERE qs.database_id = DB_ID()
ORDER BY avg_elapsed_time DESC OPTION (RECOMPILE);
-- This gives you some interesting information about the variability in the
execution time of your cached stored procedures, which is useful for tuning
```

Top Cached SPs By Total Worker time (SQL 2008). Worker time relates to CPU cost

```
SELECT TOP(25) p.name AS [SP Name], qs.total_worker_time AS [TotalWorkerTime],
qs.total_worker_time/qs.execution_count AS [AvgWorkerTime], qs.execution_count,
ISNULL(qs.execution_count/DATEDIFF(Second, qs.cached_time, GETDATE()), 0) AS
[Calls/Second],
qs.total_elapsed_time, qs.total_elapsed_time/qs.execution_count
AS [avg_elapsed_time], qs.cached_time
FROM sys.procedures AS p WITH (NOLOCK)
INNER JOIN sys.dm_exec_procedure_stats AS qs WITH (NOLOCK)
ON p.[object_id] = qs.[object_id]
WHERE qs.database_id = DB_ID()
ORDER BY qs.total_worker_time DESC OPTION (RECOMPILE);
-- This helps you find the most expensive cached stored procedures from a CPU
perspective
-- You should look at this if you see signs of CPU pressure
```

Top Cached SPs By Total Logical Reads (SQL 2008). Logical reads relate to memory pressure

```
SELECT TOP(25) p.name AS [SP Name], qs.total_logical_reads AS [TotalLogicalReads],
qs.total_logical_reads/qs.execution_count AS [AvgLogicalReads],qs.execution_count,
ISNULL(qs.execution_count/DATEDIFF(Second, qs.cached_time, GETDATE()), 0) AS
[Calls/Second],
qs.total_elapsed_time, qs.total_elapsed_time/qs.execution_count
AS [avg_elapsed_time], qs.cached_time
FROM sys.procedures AS p WITH (NOLOCK)
INNER JOIN sys.dm_exec_procedure_stats AS qs WITH (NOLOCK)
ON p.[object_id] = qs.[object_id]
WHERE qs.database_id = DB_ID()
ORDER BY qs.total_logical_reads DESC OPTION (RECOMPILE);
-- This helps you find the most expensive cached stored procedures from a memory
perspective
-- You should look at this if you see signs of memory pressure
```

Top Cached SPs By Total Physical Reads (SQL 2008). Physical reads relate to disk I/O pressure

```
SELECT TOP(25) p.name AS [SP Name],qs.total_physical_reads AS [TotalPhysicalReads],
qs.total_physical_reads/qs.execution_count AS [AvgPhysicalReads],
qs.execution_count,
qs.total_logical_reads,qs.total_elapsed_time,
qs.total_elapsed_time/qs.execution_count
AS [avg_elapsed_time], qs.cached_time
FROM sys.procedures AS p WITH (NOLOCK)
INNER JOIN sys.dm_exec_procedure_stats AS qs WITH (NOLOCK)
ON p.[object_id] = qs.[object_id]
WHERE qs.database_id = DB_ID()
AND qs.total_physical_reads > 0
ORDER BY qs.total_physical_reads DESC, qs.total_logical_reads DESC OPTION
(RECOMPILE);
-- This helps you find the most expensive cached stored procedures from a read I/O
perspective
-- You should look at this if you see signs of I/O pressure or of memory pressure
```

Top Cached SPs By Total Logical Writes (SQL 2008).

```
-- Logical writes relate to both memory and disk I/O pressure
SELECT TOP(25) p.name AS [SP Name], qs.total_logical_writes AS
[TotalLogicalWrites],
qs.total_logical_writes/qs.execution_count AS [AvgLogicalWrites],
qs.execution_count,
ISNULL(qs.execution_count/DATEDIFF(Second, qs.cached_time, GETDATE()), 0) AS
[Calls/Second],
qs.total_elapsed_time, qs.total_elapsed_time/qs.execution_count AS
[avg_elapsed_time],
qs.cached_time
FROM sys.procedures AS p WITH (NOLOCK)
INNER JOIN sys.dm_exec_procedure_stats AS qs WITH (NOLOCK)
ON p.[object_id] = qs.[object_id]
WHERE qs.database_id = DB_ID()
ORDER BY qs.total_logical_writes DESC OPTION (RECOMPILE);
-- This helps you find the most expensive cached stored procedures from a write I/O
perspective
-- You should look at this if you see signs of I/O pressure or of memory pressure
```

Lists the top statements by average input/output usage for the current database

```
SELECT TOP(50) OBJECT_NAME(qt.objectid) AS [SP Name],
(qs.total_logical_reads + qs.total_logical_writes) /qs.execution_count AS [Avg IO],
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 WITH (NOLOCK)
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS qt
WHERE qt.[dbid] = DB_ID()
ORDER BY [Avg IO] DESC OPTION (RECOMPILE);
-- Helps you find the most expensive statements for I/O by SP
```

Possible Bad NC Indexes (writes > reads)

```
SELECT OBJECT_NAME(s.[object_id]) AS [Table Name], i.name AS [Index Name],
i.index_id,
user_updates AS [Total Writes], user_seeks + user_scans + user_lookups AS [Total
Reads],
user_updates - (user_seeks + user_scans + user_lookups) AS [Difference]
FROM sys.dm_db_index_usage_stats AS s WITH (NOLOCK)
INNER JOIN sys.indexes AS i WITH (NOLOCK)
ON s.[object_id] = i.[object_id]
AND i.index_id = s.index_id
WHERE OBJECTPROPERTY(s.[object_id],'IsUserTable') = 1
AND s.database_id = DB_ID()
AND user_updates > (user_seeks + user_scans + user_lookups)
AND i.index_id > 1
ORDER BY [Difference] DESC, [Total Writes] DESC, [Total Reads] ASC OPTION
(RECOMPILE);
-- Look for indexes with high numbers of writes and zero or very low numbers of
reads
-- Consider your complete workload
-- Investigate further before dropping an index!
```

Missing Indexes current database by Index Advantage

```
SELECT user_seeks * avg_total_user_cost * (avg_user_impact * 0.01) AS
[index_advantage],
migs.last_user_seek, mid.[statement] AS [Database.Schema.Table],
mid.equality_columns, mid.inequality_columns, mid.included_columns,
migs.unique_compiles, migs.user_seeks, migs.avg_total_user_cost,
migs.avg_user_impact
FROM sys.dm_db_missing_index_group_stats AS migs WITH (NOLOCK)
INNER JOIN sys.dm_db_missing_index_groups AS mig WITH (NOLOCK)
ON migs.group_handle = mig.index_group_handle
INNER JOIN sys.dm_db_missing_index_details AS mid WITH (NOLOCK)
ON mig.index_handle = mid.index_handle
WHERE mid.database_id = DB_ID() -- Remove this to see for entire instance
ORDER BY index_advantage DESC OPTION (RECOMPILE);

-- Look at index advantage, last user seek time, number of user seeks to help
determine source and importance
-- SQL Server is overly eager to add included columns, so beware
-- Do not just blindly add indexes that show up from this query!!!
```

Find missing index warnings for cached plans in the current database

```
-- Note: This query could take some time on a busy instance
SELECT TOP(25) OBJECT_NAME(objectid) AS [ObjectName],
               query_plan, cp.objtype, cp.usecounts
FROM sys.dm_exec_cached_plans AS cp WITH (NOLOCK)
CROSS APPLY sys.dm_exec_query_plan(cp.plan_handle) AS qp
WHERE CAST(query_plan AS NVARCHAR(MAX)) LIKE N'%MissingIndex%'
AND dbid = DB_ID()
ORDER BY cp.usecounts DESC OPTION (RECOMPILE);
-- Helps you connect missing indexes to specific stored procedures
-- This can help you decide whether to add them or not
```

Breaks down buffers used by current database by object (table, index) in the buffer cache

```
-- Note: This query could take some time on a busy instance
SELECT OBJECT_NAME(p.[object_id]) AS [ObjectName],
p.index_id, COUNT(*)/128 AS [Buffer size(MB)],  COUNT(*) AS [BufferCount],
p.data_compression_desc AS [CompressionType], a.type_desc, p.[rows]
FROM sys.allocation_units AS a WITH (NOLOCK)
INNER JOIN sys.dm_os_buffer_descriptors AS b WITH (NOLOCK)
ON a.allocation_unit_id = b.allocation_unit_id
INNER JOIN sys.partitions AS p WITH (NOLOCK)
ON a.container_id = p.partition_id
WHERE b.database_id = CONVERT(int,DB_ID())
AND p.[object_id] > 100
GROUP BY p.[object_id], p.index_id, p.data_compression_desc, a.type_desc, p.[rows]
ORDER BY [BufferCount] DESC OPTION (RECOMPILE);
-- Tells you what tables and indexes are using the most memory in the buffer cache
```

Get Table names, row counts, and compression status for clustered index or heap

```
SELECT OBJECT_NAME(object_id) AS [ObjectName],
SUM(Rows) AS [RowCount], data_compression_desc AS [CompressionType]
FROM sys.partitions WITH (NOLOCK)
WHERE index_id < 2 --ignore the partitions from the non-clustered index if any
AND OBJECT_NAME(object_id) NOT LIKE N'sys%'
AND OBJECT_NAME(object_id) NOT LIKE N'queue_%'
AND OBJECT_NAME(object_id) NOT LIKE N'filestream_tombstone%'
AND OBJECT_NAME(object_id) NOT LIKE N'fulltext%'
AND OBJECT_NAME(object_id) NOT LIKE N'ifts_comp_fragment%'
GROUP BY object_id, data_compression_desc
ORDER BY SUM(Rows) DESC OPTION (RECOMPILE);
-- Gives you an idea of table sizes, and possible data compression opportunities
```

When were Statistics last updated on all indexes?

```
SELECT o.name, i.name AS [Index Name],
      STATS_DATE(i.[object_id], i.index_id) AS [Statistics Date],
      s.auto_created, s.no_recompute, s.user_created, st.row_count
FROM sys.objects AS o WITH (NOLOCK)
INNER JOIN sys.indexes AS i WITH (NOLOCK)
ON o.[object_id] = i.[object_id]
INNER JOIN sys.stats AS s WITH (NOLOCK)
ON i.[object_id] = s.[object_id]
AND i.index_id = s.stats_id
INNER JOIN sys.dm_db_partition_stats AS st WITH (NOLOCK)
ON o.[object_id] = st.[object_id]
AND i.[index_id] = st.[index_id]
WHERE o.[type] = 'U'
ORDER BY STATS_DATE(i.[object_id], i.index_id) ASC OPTION (RECOMPILE);
-- Helps discover possible problems with out-of-date statistics
-- Also gives you an idea which indexes are the most active
```

Get fragmentation info for all indexes above a certain size in the current database

```
-- Note: This could take some time on a very large database
SELECT DB_NAME(database_id) AS [Database Name], OBJECT_NAME(ps.OBJECT_ID) AS
[Object Name],
i.name AS [Index Name], ps.index_id, index_type_desc,
avg_fragmentation_in_percent, fragment_count, page_count
FROM sys.dm_db_index_physical_stats(DB_ID(),NULL, NULL, NULL ,'LIMITED') AS ps
INNER JOIN sys.indexes AS i WITH (NOLOCK)
ON ps.[object_id] = i.[object_id]
AND ps.index_id = i.index_id
WHERE database_id = DB_ID()
AND page_count > 1500
ORDER BY avg_fragmentation_in_percent DESC OPTION (RECOMPILE);
-- Helps determine whether you have fragmentation in your relational indexes and
how effective your index maintenance strategy is
```

Index Read/Write stats (all tables in current DB) ordered by Reads

```
SELECT OBJECT_NAME(s.[object_id]) AS [ObjectName], i.name AS [IndexName],
i.index_id,
           user_seeks + user_scans + user_lookups AS [Reads], s.user_updates AS
[Writes],
           i.type_desc AS [IndexType], i.fill_factor AS [FillFactor]
FROM sys.dm_db_index_usage_stats AS s WITH (NOLOCK)
INNER JOIN sys.indexes AS i WITH (NOLOCK)
ON s.[object_id] = i.[object_id]
WHERE OBJECTPROPERTY(s.[object_id],'IsUserTable') = 1
AND i.index_id = s.index_id
AND s.database_id = DB_ID()
ORDER BY user_seeks + user_scans + user_lookups DESC OPTION (RECOMPILE); -- Order
by reads
-- Show which indexes in the current database are most active for Reads
```

Index Read/Write stats (all tables in current DB) ordered by Writes

```
SELECT OBJECT_NAME(s.[object_id]) AS [ObjectName], i.name AS [IndexName],
i.index_id,
           s.user_updates AS [Writes], user_seeks + user_scans + user_lookups AS
[Reads],
           i.type_desc AS [IndexType], i.fill_factor AS [FillFactor]
FROM sys.dm_db_index_usage_stats AS s WITH (NOLOCK)
INNER JOIN sys.indexes AS i WITH (NOLOCK)
ON s.[object_id] = i.[object_id]
WHERE OBJECTPROPERTY(s.[object_id],'IsUserTable') = 1
AND i.index_id = s.index_id
AND s.database_id = DB_ID()
ORDER BY s.user_updates DESC OPTION (RECOMPILE);
         -- Order by writes
-- Show which indexes in the current database are most active for Writes
```

Creating a Role and assigning to user

```
CREATE ROLE [Developer] AUTHORIZATION db_securityadmin;

GRANT CREATE PROCEDURE TO [Developer]; GRANT SELECT, INSERT, UPDATE, DELETE, ALTER,
EXECUTE, VIEW DEFINITION ON SCHEMA::dbo TO [Developer]

GRANT CREATE VIEW TO [Developer]

EXEC sp_addrolemember @rolename = 'Developer', @membername = 'RPM_User';
```

SQL_Login_Password_Expiry_Notification

```
USE [master]
GO

/****** Object:  Stored Procedure dbo.usp_SQL_Login_Password_Expiry_Notification
******/
SET ANSI_NULLS ON
GO
```

```
SET QUOTED_IDENTIFIER ON
GO

-- =============================================
--     Description :This store procedure will send a proactivily mail -- about the
SQL login name & their exprìry date for which login which logins was applied for
enforce the password policy
-- =============================================
CREATE PROCEDURE dbo.usp_SQL_Login_Password_Expriy_Notification
      AS
BEGIN
      DECLARE @Days INT
      DECLARE @AlertMessage VARCHAR(500)
      DECLARE @Count INT
      DECLARE @intFlag INT
      SET @intFlag = 1
      set @Count=(SELECT COUNT(name)FROM master.sys.sql_logins where
is_expiration_checked <> 0)
      WHILE (@intFlag <=@Count)
            BEGIN
                  DECLARE @name NVARCHAR(50)
                  DECLARE @ExpDate DATETIME

                  SET @name =(      SELECT name FROM (select ROW_NUMBER () OVER
(ORDER BY NAME ) AS 'SrNo',
                                          name FROM master.sys.sql_logins where
is_expiration_checked <> 0  ) AS pp
                                          WHERE SrNo=@intFlag
                                    )
                  SET @ExpDate=(SELECT GETDATE()+ CAST((select loginproperty(@name
,'DaysUntilExpiration')) AS int))
                  SET @AlertMessage = @name +    '  SQL login will exprie on  '   +
cast(@ExpDate as varchar(50))

            IF @ExpDate=GETDATE()+2
                  BEGIN
                        EXEC msdb.dbo.sp_send_dbmail
                        @profile_name = 'Mail_Profile',
                        @recipients = 'pandeyl@ninds.nih.gov',
                        @body = @AlertMessage,
                        @subject = 'password policy Exipration' ;
                  end
            ELSE
            begin
            PRINT 'Not Record Found'
            end
            SET @intFlag = @intFlag + 1
  END
END

GO
```

Compression of all objects

```
use DBMonitor
go

if exists (select * from sys.objects where object_id = object_id(N'[dbo].[sp_Compress_All_Objects]')
and type in (N'P', N'PC'))
   drop procedure [dbo].[sp_Compress_All_Objects]
go

use DBMonitor
go

/****** Object: StoredProcedure [dbo].[usp_Compress_All_Objects] Script Date: 12/12/2011 07:45:00
******/
set ansi_nulls on
go

set quoted_identifier on
go

/*********************************************************************
                  Compresses all objects at the same level
   Assumes the same level of compression for all partitions, uses partition #1 to make that decision
   Does not take into account scan/update ratios to determine best candidates for compression
   Does not take into account table/index sizes for determining best candidates for compression
   Uses Offline rebuild operation
   Compresses all indexes regardless of their current compression state
   SQL 2008 Enterprise Edition or higher
      @DB = Valid Database Name
   @CompressionType = Compression Type (PAGE, ROW, NONE)
   @TablesOrIndexes = Indicates if tables or indexes are to be compressed, mutually exclusive (Tables,
Indexes)
   @ExecCompression = Indicates whether the compression statement is run or if dynamic SQL is
created as output (true, false)
   examples:
      exec DBMonitor..sp_Compress_All_Objects
         @DB = 'sysutility_mdw',
         @CompressionType = 'PAGE',
         @TablesOrIndexes = 'Tables',
         @ExecCompression = 'true'

*********************************************************************/
create procedure [dbo].[usp_Compress_All_Objects]
   (
   @DB varchar(256) = NULL,
   @CompressionType varchar(8) = 'PAGE',
   @TablesOrIndexes varchar(8) = 'Tables',
   @ExecCompression varchar(8) = 'true'
   )

   as

   declare @SQLCmd as varchar(8000)
   declare @SQLCmdBuild as nvarchar(4000)
   declare @ErrorTxt as varchar(128)
   declare @SQLEdition as varchar(64)
   declare @SQLVersion varchar(128)
   declare @EndTime as datetime
   declare @CompressionTypeNum as smallint
   declare @UnhandledError as int
   declare @n as int

   declare @SchemaName as varchar(1024)
   declare @TableName as varchar(1024)
   declare @IndexName as varchar(1024)

   --Drop Temp Tables
   if exists (select * from tempdb.sys.objects where name = '##ListOfTables')
      drop table ##ListOfTables
   if exists (select * from tempdb.sys.objects where name = N'##ListofPartitionsAndCompression')
      drop table ##ListofPartitionsAndCompression
   if exists (select * from tempdb.sys.objects where name = N'##TableList')
      drop table ##TableList
   if exists (select * from tempdb.sys.objects where name = N'##ListOfIndexes')
      drop table ##ListOfIndexes

   set @CompressionType = upper(@CompressionType)

   --Determine Version and Edition to check for support of compression
   set @SQLVersion = cast(ServerProperty('ProductVersion') as varchar)
   set @SQLEdition = cast(ServerProperty('Edition') as varchar)
   set @SQLEdition =
      case
         when left(@SQLEdition, 11) = 'Data Center' then 'Data Center'
         when left(@SQLEdition, 10) = 'Enterprise' then 'Enterprise'
         when left(@SQLEdition, 9) = 'Developer' then 'Developer'
         when left(@SQLEdition, 8) = 'Standard' then 'Standard'
      end

   set @CompressionTypeNum =
      case
```

```
        when @CompressionType = 'NONE' then 0
        when @CompressionType = 'ROW' then 1
        when @CompressionType = 'PAGE' then 2
    end

--Verify that database name is valid
if not exists (select name from sys.databases where name = @DB)
    begin
        set @ErrorTxt = 'The supplied Database name is incorrect'
        RaisError (@ErrorTxt, 16, 1)
        return
    end

--Check that selected compression type is valid
if upper(@CompressionType) not in ('ROW', 'PAGE', 'NONE')
    begin
        set @ErrorTxt = 'Compression Type must = "ROW", "PAGE", or "NONE"'
        RaisError (@ErrorTxt, 16, 1)
        return
    end

--Check that SQL supports compression by edition
if @SQLEdition not in ('Data Center', 'Enterprise', 'Developer')
    begin
        set @ErrorTxt = 'The edition of SQL must be Enterprise or Developer to support Compression'
        RaisError (@ErrorTxt, 16, 1)
        return
    end

--Check that SQL supports compression by version (2008, 2008 R2, or 2012)
if left(@SQLVersion, 2) not in ('10', '11')
    begin
        set @ErrorTxt = 'Compression is only supported in SQL 2008 RTM and later'
        RaisError (@ErrorTxt, 16, 1)
        return
    end

--Make sure that Tables or Indexes are selected
if lower(@TablesOrIndexes) not in ('tables', 'indexes')
    begin
        set @ErrorTxt = 'You must select either "Tables" or "Indexes" for compression'
        RaisError (@ErrorTxt, 16, 1)
        return
    end

--Check for debug mode
```

```
if lower(@ExecCompression) not in ('true', 'false')
    begin
        set @ErrorTxt = 'You must select either "true" or "false" for @ExecCompression'
        RaisError (@ErrorTxt, 16, 1)
        return
    end

--Start Compression of TABLES
if @TablesOrIndexes = 'Tables'
    begin
        --Build Temp Tables of table names
        set @SQLCmdBuild = 'use ' + @DB + char(13) +
        'select ss.[name] as [Schema Name], so.name as [Table Name], so.object_id as [Object ID]
            into ##ListOfTables
            from sys.objects so
            inner join sys.schemas as ss on so.schema_id = ss.schema_id
            order by ss.name, so.name'
        exec sp_executesql @SQLCmdBuild

        set @SQLCmdBuild = 'use ' + @DB + char(13) +
        'select lt.*, sp.partition_number as [Partition Number], sp.data_compression_desc as
[Compression]
            into ##ListofPartitionsAndCompression
            from ##ListOfTables lt
            inner join sys.partitions sp on sp.object_id = lt.[Object ID]'
        exec sp_executesql @SQLCmdBuild

        --Delete unwanted data
        delete from ##ListofPartitionsAndCompression where [Partition Number] <> 1
        delete from ##ListofPartitionsAndCompression where Compression = @CompressionType
        delete from ##ListofPartitionsAndCompression where [Schema Name] = 'sys'

        --Remove Duplicates (based on unique schema name)
        select distinct ([object id]), [Schema Name], [Table Name], [Partition Number], [Compression]
            into ##TableList from ##ListofPartitionsAndCompression
        drop table ##ListofPartitionsAndCompression

        --Open cursor to begin compression operations
        declare curTable cursor
            for select [Schema Name], [Table Name] from ##TableList

        open curTable
        fetch next from curTable into @SchemaName, @TableName

        while @@fetch_status <> -1
            begin
```

```
          set @SQLCmd = 'alter table ' + quotename(@DB) + '.' + quotename(@SchemaName) + '.'
             + quotename(@TableName) + ' rebuild partition = all with (data_compression = ' +
             @CompressionType + ')'
          begin try
             if @ExecCompression = 'true'
                begin
                   print 'compressing ' + @SchemaName + '.' + @TableName
                   exec (@SQLCmd)
                end
                else print @SQLCmd
          end try
          begin catch
             set @UnhandledError = @UnhandledError + 1
          end catch
          fetch next from curTable into @SchemaName, @TableName
       end

    close curTable
    deallocate curtable

    drop table ##ListOfTables
    drop table ##TableList
    return
 end --Compression of TABLES

--Start Compression of INDEXES
if @TablesOrIndexes = 'Indexes'
 begin
    --Build Temp Tables of index names
    set @SQLCmdBuild = 'use ' + @DB + char(13) +
       'select si.name as [Index], object_name(si.object_id) as [Table]
          into ##ListOfTables
          from sys.indexes si
          where name is not null
          order by [table]'
    exec sp_executesql @SQLCmdBuild

    set @SQLCmdBuild = 'use ' + @DB + char(13) +
       'select schema_name(schema_id) as [Schema], lt.*
          into ##ListOfIndexes
          from sys.tables st
          inner join ##ListOfTables as lt on lt.[Table] = st.[name]
          order by [schema], [table], [index]'
    exec sp_executesql @SQLCmdBuild

    --Open cursor to begin compression operations
```

```
        declare curIndexes cursor
           for select [Schema], [Index], [Table] from ##ListOfIndexes

        open curIndexes
        fetch next from curIndexes into @SchemaName, @IndexName, @TableName

        while @@fetch_status <> -1
           begin
              set @SQLCmd = 'alter index ' + quotename(@DB) + '.' + quotename(@SchemaName) + '.'
                 + quotename(@TableName) + + quotename(@IndexName) +
                 ' rebuild partition = all with (data_compression = ' + @CompressionType + ')'
              begin try
                 if @ExecCompression = 'true'
                    begin
                       print 'compressing ' + @SchemaName + '.' + @TableName + '.' + @IndexName
                       exec (@SQLCmd)
                    end
                    else print @SQLCmd
              end try
              begin catch
                 set @UnhandledError = @UnhandledError + 1
              end catch
              fetch next from curIndexes into @SchemaName, @IndexName, @TableName
           end

        close curIndexes
        deallocate curIndexes
     end --Compression of INDEXES
```

When was my Databases last accessed?

```
SELECT DatabaseName, MAX(LastAccessDate) LastAccessDate
FROM
(SELECT
DB_NAME(database_id) DatabaseName
, last_user_seek
, last_user_scan
, last_user_lookup
, last_user_update
FROM sys.dm_db_index_usage_stats) AS PivotTable
UNPIVOT
(LastAccessDate FOR last_user_access IN
(last_user_seek
, last_user_scan
, last_user_lookup
, last_user_update)
) AS UnpivotTable
GROUP BY DatabaseName
```

```
HAVING DatabaseName NOT IN ('master', 'tempdb', 'model', 'msdb')ORDER BY 2
```

Find Resource Usage by Application

```
SELECT
     CPU             = SUM(cpu_time)
    ,WaitTime        = SUM(total_scheduled_time)
    ,ElapsedTime     = SUM(total_elapsed_time)
    ,Reads           = SUM(num_reads)
    ,Writes          = SUM(num_writes)
    ,Connections     = COUNT(1)
    ,Program         = program_name
    ,LoginName       = ses.login_name
FROM sys.dm_exec_connections con
LEFT JOIN sys.dm_exec_sessions ses
    ON ses.session_id = con.session_id
GROUP BY program_name, ses.login_name
ORDER BY cpu DESC
```

Find Long Running Queries

```
SELECT TOP 10
ObjectName          = OBJECT_NAME(qt.objectid)
,DiskReads          = qs.total_physical_reads -- The worst reads, disk reads
,MemoryReads        = qs.total_logical_reads  --Logical Reads are memory reads
,Executions         = qs.execution_count
,AvgDuration        = qs.total_elapsed_time / qs.execution_count
,CPUTime            = qs.total_worker_time
,DiskWaitAndCPUTime = qs.total_elapsed_time
,MemoryWrites       = qs.max_logical_writes
,DateCached         = qs.creation_time
,DatabaseName       = DB_Name(qt.dbid)
FROM sys.dm_exec_query_stats AS qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) AS qt
WHERE qt.dbid = db_id() -- Filter by current database
ORDER BY qs.total_elapsed_time DESC
```

Email output report from a Query

```
EXEC msdb.dbo.sp_send_dbmail
    --@profile_name = 'DBA',
    @recipients = 'pandeyl@ninds.nih.gov',
    @query = 'EXEC usp_LogCheckUp' ,
    @subject = 'Log Check Up',
    @attach_query_result_as_file = 1 ;
```

Find database restore history

```
USE msdb ;
SELECT
DISTINCT
        DBRestored = destination_database_name ,
        RestoreDate = restore_date ,
        SourceDB = b.database_name ,
        BackupDate = backup_start_date
FROM    RestoreHistory h
        JOIN MASTER..sysdatabases sd ON sd.name = h.destination_database_name
        INNER JOIN BackupSet b ON h.backup_set_id = b.backup_set_id
        INNER JOIN BackupFile f ON f.backup_set_id = b.backup_set_id
GROUP BY destination_database_name ,
        restore_date ,
        b.database_name ,
        backup_start_date
ORDER BY RestoreDate DESC
GO
```

Useful sysprocesses select

```
SELECT spid
,blocked
,DB_NAME(sp.dbid) as DBName
,program_name
,waitresource
,lastwaittype
,sp.loginame
,sp.hostname
,a.[Text] as [TextData]
,SUBSTRING(A.text, sp.stmt_start / 2,
 (CASE
WHEN sp.stmt_end = -1
 THEN DATALENGTH(A.text)
 ELSE sp.stmt_end
 END - sp.stmt_start
 )/2) AS [current_cmd]
FROM sys.sysprocesses sp
 OUTER APPLY sys.dm_exec_sql_text (sp.sql_handle) as A
WHERE
 spid > 50
ORDER BY
 blocked DESC
,DB_NAME(sp.dbid) ASC
,a.[text]
```

Operators With Jobs and Notifications

```
-------------------------------------------------------
-- Sysoperators With Jobs
```

```
--------------------------------------------------------
--This script will list the operators that have jobs
--that are configured to notify them with alerts.
--The output indicates if the operator is enabled
--or not and also if the job is enabled.
-- THIS SCRIPT DOES NOT WORK FOR SQL 2000
--------------------------------------------------------
SET NOCOUNT ON

SELECT o.[id]
 ,o.[name] AS Operator
 ,CASE WHEN o.[enabled] =1 THEN 'YES' ELSE 'NO' END AS OperatorEnabled
 ,o.[email_address]
 ,o.[last_email_date]
 ,o.[last_email_time]
 ,CASE WHEN j.[enabled] =1 THEN 'YES' ELSE 'NO' END AS JobEnabled
 ,j.[name] AS JobName
 ,j.[description] AS JobDescription
FROM
[msdb].[dbo].[sysoperators] o INNER JOIN
[msdb].[dbo].[sysjobs_view] j ON o.[id] = j.[notify_email_operator_id]
WHERE
j.[notify_email_operator_id] != 0
```

Find the explicit permissions granted or denied in a database

```
SELECT
  perms.state_desc AS State,
  permission_name AS [Permission],
  obj.name AS [on Object],
  dPrinc.name AS [to User Name],
   sPrinc.name AS [who is Login Name]
FROM sys.database_permissions AS perms
JOIN sys.database_principals AS dPrinc
     ON perms.grantee_principal_id = dPrinc.principal_id
JOIN sys.objects AS obj
     ON perms.major_id = obj.object_id
LEFT OUTER JOIN sys.server_principals AS sPrinc
     ON dPrinc.sid = sPrinc.sid
```

Find the members of the server roles

```
SELECT sRole.name AS [Server Role Name] , sPrinc.name AS [Members]
FROM sys.server_role_members AS sRo
JOIN sys.server_principals AS sPrinc
     ON sRo.member_principal_id = sPrinc.principal_id
JOIN sys.server_principals AS sRole
     ON sRo.role_principal_id = sRole.principal_id;
```

Find the members of the database roles

```
SELECT dRole.name AS [Database Role Name], dPrinc.name AS [Members]
FROM sys.database_role_members AS dRo
JOIN sys.database_principals AS dPrinc
     ON dRo.member_principal_id = dPrinc.principal_id
JOIN sys.database_principals AS dRole
     ON dRo.role_principal_id = dRole.principal_id;
```

Clean up the backup history

```
EXEC sp_delete_backuphistory '1/1/2009'
--would delete backup data older than Janunary 1, 2009.
```

Returns the Current Error Log

```
Declare @ErrorLog Table (LogID int identity(1, 1) not null primary key,
        LogDate datetime null,
        ProcessInfo nvarchar(100) null,
        LogText nvarchar(4000) null)

Insert Into @ErrorLog (LogDate, ProcessInfo, LogText)
Exec master..xp_readerrorlog

Select *
From @ErrorLog
Where CharIndex('Backup', ProcessInfo) = 0
Order By LogID Desc
```

Returns Info and Stats about IO Stall

```
Declare @Counter int
Declare @ErrorLog Table (LogID int identity(1, 1) not null primary key,
                        LogDate datetime null,
                        ProcessInfo nvarchar(100) null,
                        LogText nvarchar(max) null)

Set @Counter = 0

While @Counter < 2
  Begin
    Insert Into @ErrorLog (LogDate, ProcessInfo, LogText)
    Exec master..xp_readerrorlog @Counter

    Set @Counter = @Counter + 1
  End

Select Count(LogText), Sum(Cast(Left(Right(LogText, Len(LogText) - 27),
CharIndex(space(1), Right(LogText, Len(LogText) - 27))) as int))
From @ErrorLog
Where CharIndex('I/O requests taking longer than 15 seconds to complete', LogText)
> 0

Select Cast(Convert(varchar, LogDate, 110) as datetime) As dPerDay,
DatePart(hour, LogDate) As PerHour,
DatePart(minute, LogDate) As PerMinute,
Convert(varchar, LogDate, 110) As PerDay, Count(LogText) IOWarningsLogged,
TotalIOSlowDowns = Sum(Cast(Left(Right(LogText, Len(LogText) - 27),
CharIndex(space(1), Right(LogText, Len(LogText) - 27))) as int))
From @ErrorLog
Where CharIndex('I/O requests taking longer than 15 seconds to complete', LogText)
> 0
Group By Convert(varchar, LogDate, 110), DatePart(hour, LogDate), DatePart(minute,
LogDate)
Order By dPerDay desc, PerHour desc, PerMinute desc
```

When Was Database Integrity Last Checked

```
Declare @DBs Table (
       Id int identity(1,1) primary key,
       ParentObject varchar(255),
       Object varchar(255),
       Field varchar(255),
       Value varchar(255)
)

Insert Into @DBs (ParentObject, Object, Field, Value)
Exec sp_msforeachdb N'DBCC DBInfo(''?'') With TableResults;';

Insert Into @DBs (ParentObject, Object, Field, Value)
Select 'Final Record', 'Final Record', 'dbi_dbname', 'Final Record';

With DBNames (Id, Field, Value, DBID)
As (Select Id, Field, Value,
     ROW_NUMBER() OVER (PARTITION BY Field ORDER BY ID)
    From @DBs
    Where Field = 'dbi_dbname')
, LastDBCC (Id, Field, Value)
As (Select Id, Field, Value
    From @DBs
    Where Field = 'dbi_dbccLastKnownGood')
Select Distinct D1.Value, L.Value
From LastDBCC L
Inner Join DBNames D1 On L.Id > D1.Id
Inner Join DBNames D2 On L.Id < D2.Id And D2.DBID = D1.DBID + 1;
```

Waits Over the Last one Minute

```
Declare @Waits Table (
    WaitID int identity(1, 1) not null primary key,
    wait_type nvarchar(60),
    wait_time_s decimal(12, 2));

WITH Waits AS
(SELECT wait_type, wait_time_ms / 1000. AS wait_time_s,
    100. * wait_time_ms / SUM(wait_time_ms) OVER() AS pct,
    ROW_NUMBER() OVER(ORDER BY wait_time_ms DESC) AS rn
FROM sys.dm_os_wait_stats
WHERE wait_type NOT IN( 'SLEEP_TASK', 'BROKER_TASK_STOP',
  'SQLTRACE_BUFFER_FLUSH', 'CLR_AUTO_EVENT', 'CLR_MANUAL_EVENT',
  'LAZYWRITER_SLEEP')) -- filter out additional irrelevant waits
Insert Into @Waits (wait_type, wait_time_s)
SELECT W1.wait_type,
  CAST(W1.wait_time_s AS DECIMAL(12, 2)) AS wait_time_s
FROM Waits AS W1
INNER JOIN Waits AS W2
ON W2.rn <= W1.rn
GROUP BY W1.rn, W1.wait_type, W1.wait_time_s, W1.pct
HAVING SUM(W2.pct) - W1.pct < 95; -- percentage threshold

WaitFor Delay '0:01:00';

WITH Waits AS
(SELECT wait_type, wait_time_ms / 1000. AS wait_time_s,
    100. * wait_time_ms / SUM(wait_time_ms) OVER() AS pct,
    ROW_NUMBER() OVER(ORDER BY wait_time_ms DESC) AS rn
FROM sys.dm_os_wait_stats
WHERE wait_type NOT IN( 'SLEEP_TASK', 'BROKER_TASK_STOP',
  'SQLTRACE_BUFFER_FLUSH', 'CLR_AUTO_EVENT', 'CLR_MANUAL_EVENT',
  'LAZYWRITER_SLEEP')) -- filter out additional irrelevant waits
Insert Into @Waits (wait_type, wait_time_s)
SELECT W1.wait_type,
  CAST(W1.wait_time_s AS DECIMAL(12, 2)) AS wait_time_s
FROM Waits AS W1
INNER JOIN Waits AS W2
ON W2.rn <= W1.rn
GROUP BY W1.rn, W1.wait_type, W1.wait_time_s, W1.pct
HAVING SUM(W2.pct) - W1.pct < 95; -- percentage threshold

Select wait_type, MAX(wait_time_s) - MIN(wait_time_s) WaitDelta
From @Waits
Group By wait_Type
Order By WaitDelta Desc
```

All SQL Version info in one function

```
--#region drop if exists
if exists (select 1 from INFORMATION_SCHEMA.ROUTINES where ROUTINE_NAME =
'udf_sqlversioninfo' and ROUTINE_SCHEMA='dbo' and ROUTINE_TYPE='FUNCTION')
drop function [dbo].[udf_sqlversioninfo];
go
--#endregion

--#region create function udf_sqlversioninfo
create function dbo.udf_sqlversioninfo
()
/*
 name udf_sqlversioninfo
 returns   @productinfo table
 purpose Returns SQL Version info in a table
 parameters (none)
*/
returns @productinfo table(ProductVersion sysname, ProductValue numeric(18,7),
ProductName varchar(16), ProductLevel varchar(8), Major int, Minor int, Build int,
BuildVersion int, Edition sysname, EngineEdition varchar(16), LicenseType sysname)
as
```

```sql
begin
insert into @productinfo(ProductVersion, ProductLevel, Major, Minor, Build,
BuildVersion, Edition, LicenseType)
select cast(serverproperty('Productversion') as sysname)
, cast(serverproperty('ProductLevel') as sysname)
, parsename(cast(serverproperty('Productversion') as sysname),4)
, parsename(cast(serverproperty('Productversion') as sysname),3)
, parsename(cast(serverproperty('Productversion') as sysname),2)
, parsename(cast(serverproperty('Productversion') as sysname),1)
, cast(serverproperty('Edition') as sysname)
, cast(serverproperty('LicenseType') as sysname)

update @productinfo
set Productvalue = cast(
parsename(cast(serverproperty('Productversion') as sysname),4)
+ '.'
+ parsename(cast(serverproperty('Productversion') as sysname),3)
+ parsename(cast(serverproperty('Productversion') as sysname),2)
+ parsename(cast(serverproperty('Productversion') as sysname),1)
 as numeric(18,7));

update @productinfo
set ProductName = 'MSSQL ' + case
when [Major] = 9 then '2005'
when Major = 10 and Minor = 0 then '2008'
when Major = 10 and Minor = 50 then '2008R2'
when Major = 11 then '2012'
else cast(Major as varchar) + '.' + cast(Minor as varchar)
end ;

update @productinfo
set EngineEdition = case cast(serverproperty('EngineEdition') as int)
when 1 then 'Personal'
when 2 then 'Standard'
when 3 then 'Enterprise'
when 4 then 'Express'
when 5 then 'Azure'
else 'unknown (' + cast(serverproperty('EngineEdition') as varchar) + ')'
end;
return;
end
go
--#endregion
--run this to get the output
select * from dbo.udf_SQLVersionInfo()
```

Task Progress

```sql
select
convert (varchar(50),(estimated_completion_time/3600000))+'hrs'+
convert (varchar(50), ((estimated_completion_time%3600000)/60000))+'min'+
convert (varchar(50), (((estimated_completion_time%3600000)%60000)/1000))+'sec'
as Estimated_Completion_Time,
status, command, db_name(database_id), percent_complete
from sys.dm_exec_requests
```

Queries that use the most IO

```
SELECT TOP 10
[Total IO] = (qs.total_logical_reads + qs.total_logical_writes)
, [Average IO] = (qs.total_logical_reads + qs.total_logical_writes) /
qs.execution_count
, qs.execution_count
, SUBSTRING (qt.text,(qs.statement_start_offset/2) + 1,
((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) + 1) AS [Individual Query]
, qt.text AS [Parent Query]
, DB_NAME(qt.dbid) AS DatabaseName
, qp.query_plan
FROM sys.dm_exec_query_stats qs
CROSS APPLY sys.dm_exec_sql_text(qs.sql_handle) as qt
CROSS APPLY sys.dm_exec_query_plan(qs.plan_handle) qp
ORDER BY [Total IO] DESC
```

Viewing tempdb Size and Growth Parameters

```
SELECT
    name AS FileName,
    size*1.0/128 AS FileSizeinMB,
    CASE max_size
        WHEN 0 THEN 'Autogrowth is off.'
        WHEN -1 THEN 'Autogrowth is on.'
        ELSE 'Log file will grow to a maximum size of 2 TB.'
    END,
    growth AS 'GrowthValue',
    'GrowthIncrement' =
        CASE
            WHEN growth = 0 THEN 'Size is fixed and will not grow.'
            WHEN growth > 0 AND is_percent_growth = 0
                THEN 'Growth value is in 8-KB pages.'
            ELSE 'Growth value is a percentage.'
        END
FROM tempdb.sys.database_files;
GO
```

Most reads or writes on data and log files

```
SELECT DB_NAME(mf.database_id) AS databaseName
,mf.physical_name
,num_of_reads
,num_of_bytes_read
,num_of_writes
,num_of_bytes_written
,size_on_disk_bytes
FROM sys.dm_io_virtual_file_stats(NULL, NULL) AS divfs
JOIN sys.master_files AS mf ON mf.database_id = divfs.database_id
AND mf.file_id = divfs.file_id
--WHERE DB_NAME(mf.database_id) = 'tempdb'
ORDER BY 1, 3 DESC
```

Find Tables without Clustered Indexes

```
Use CLIPS --database name

SELECT SCHEMA_NAME(o.schema_id) AS [schema]
      ,object_name(i.object_id ) AS [table]
    ,p.rows
    ,user_seeks
    ,user_scans
    ,user_lookups
    ,user_updates
    ,last_user_seek
    ,last_user_scan
    ,last_user_lookup
FROM sys.indexes i
      INNER JOIN sys.objects o ON i.object_id = o.object_id
    INNER JOIN sys.partitions p ON i.object_id = p.object_id AND i.index_id =
p.index_id
    LEFT OUTER JOIN sys.dm_db_index_usage_stats ius ON i.object_id = ius.object_id
AND i.index_id = ius.index_id
WHERE i.type_desc = 'HEAP'
ORDER BY rows desc
```

Update Statistics

EXEC sp_updatestats

Removing Extra Index

```
SELECT
o.name
, indexname=i.name
, i.index_id
, reads=user_seeks + user_scans + user_lookups
, writes =  user_updates
, rows = (SELECT SUM(p.rows) FROM sys.partitions p WHERE p.index_id = s.index_id
AND s.object_id = p.object_id)
, CASE
      WHEN s.user_updates < 1 THEN 100
      ELSE 1.00 * (s.user_seeks + s.user_scans + s.user_lookups) / s.user_updates
  END AS reads_per_write
, 'DROP INDEX ' + QUOTENAME(i.name)
+ ' ON ' + QUOTENAME(c.name) + '.' + QUOTENAME(OBJECT_NAME(s.object_id)) as 'drop
statement'
FROM sys.dm_db_index_usage_stats s
INNER JOIN sys.indexes i ON i.index_id = s.index_id AND s.object_id = i.object_id
INNER JOIN sys.objects o on s.object_id = o.object_id
INNER JOIN sys.schemas c on o.schema_id = c.schema_id
WHERE OBJECTPROPERTY(s.object_id,'IsUserTable') = 1
AND s.database_id = DB_ID()
AND i.type_desc = 'nonclustered'
AND i.is_primary_key = 0
AND i.is_unique_constraint = 0
AND (SELECT SUM(p.rows) FROM sys.partitions p WHERE p.index_id = s.index_id AND
s.object_id = p.object_id) > 10000
ORDER BY reads
```

Total Space Used for all databases per disk

```
begin   set nocount on
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#dbfileinfo'))  begin  drop table #dbfileinfo  end
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#logsizestats'))  begin  drop table #logsizestats  end
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#datafilestats'))  begin  drop table #datafilestats  end
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#fixeddrives'))  begin  drop table #fixeddrives  end
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#usados'))  begin  drop table #usados  end
 create table #fixeddrives  (  DriveLetter varchar(10),  MB_Free dec(20, 2)  )
 create table #datafilestats  (  DBName varchar(255),  DBId int,  FileId tinyint,
[FileGroup] tinyint,  TotalExtents dec(20, 2),  UsedExtents dec(20, 2),  [Name]
varchar(255),  [FileName] varchar(400)  )
 create table #logsizestats  (  DBName varchar(255) not null primary key
clustered,  DBId int,  LogFile real,  LogFileUsed real,  Status bit  )  create
table #dbfileinfo  (  [ServerName] varchar(255),  [DBName] varchar(65),
[LogicalFileName] varchar(400),  [UsageType] varchar (30),  [Size_MB] dec(20, 2),
[SpaceUsed_MB] dec(20, 2),  [MaxSize_MB] dec(20, 2),  [NextAllocation_MB] dec(20,
2),  [GrowthType] varchar(65),  [FileId] smallint,  [GroupId] smallint,
[PhysicalFileName] varchar(400),  [DateChecked] datetime  )
 declare @SQLString varchar(3000)  declare @MinId int  declare @MaxId int  declare
@DBName varchar(255)  declare @tblDBName table (  RowId int identity(1, 1),  DBName
varchar(255),  DBId int)
 insert into @tblDBName  (DBName,  DBId)  select [Name],  DBId  from
master..sysdatabases  where ( Status & 512 ) = 0  order by [Name]
 insert into #logsizestats  (DBName,  LogFile,  LogFileUsed,  Status)  exec ('dbcc
sqlperf(logspace) with no_infomsgs')
 update #logsizestats  set DBId = db_id(DBName)
 insert into #fixeddrives  exec master..xp_fixeddrives
 select @MinId = min(RowId),  @MaxId = max(RowId)  from @tblDBName
 while ( @MinId <= @MaxId )  begin  select @DBName = [DBName]  from @tblDBName
where RowId = @MinId
 select @SQLString = 'SELECT ServerName = @@SERVERNAME,' +  ' DBName = ''' +
@DBName +  ''',' +  ' LogicalFileName = [name],' + ' UsageType = CASE WHEN
(64&[status])=64 THEN ''Log'' ELSE ''Data'' END,' +  ' Size_MB = [size]*8/1024.00,'
+  ' SpaceUsed_MB = NULL,' +
' MaxSize_MB = CASE [maxsize] WHEN -1 THEN -1 WHEN 0 THEN [size]*8/1024.00 ELSE
maxsize/1024.00*8 END,'+
' NextExtent_MB = CASE WHEN (1048576&[status])=1048576 THEN
([growth]/100.00)*([size]*8/1024.00) WHEN [growth]=0 THEN 0 ELSE [growth]*8/1024.00
END,'+ ' GrowthType = CASE WHEN (1048576&[status])=1048576 THEN ''%'' ELSE
''Pages'' END,'+ ' FileId = [fileid],' + ' GroupId = [groupid],' +  '
PhysicalFileName= [filename],' +  ' CurTimeStamp = GETDATE()' +
'FROM [' + @DBName + ']..sysfiles'
 print @SQLString
 insert into #dbfileinfo  exec (@SQLString)
 update #dbfileinfo  set SpaceUsed_MB = Size_MB / 100.0 * (select LogFileUsed
from #logsizestats  where DBName = @DBName)  where UsageType = 'Log'  and DBName =
@DBName
  select @SQLString = 'USE [' + @DBName +  '] DBCC SHOWFILESTATS WITH NO_INFOMSGS'
  insert #datafilestats  (FileId,  [FileGroup],  TotalExtents,  UsedExtents,
[Name],  [FileName])  execute(@SQLString)
```

```
 update #dbfileinfo  set [SpaceUsed_MB] = S.[UsedExtents] * 64 / 1024.00  from
#dbfileinfo as F  inner join #datafilestats as S  on F.[FileId] = S.[FileId]  and
F.[GroupId] = S.[FileGroup]  and F.[DBName] = @DBName
 truncate table #datafilestats
 select @MinId = @MinId + 1 end
 select @@servername as servidor,  substring(A.PhysicalFileName, 1, 1) as unidad,
sum ([Size_MB]) as SqlTotalDB,  sum([SpaceUsed_MB]) as SqlTotalUsedSpaceDB,  sum ((
[Size_MB] ) - ( [SpaceUsed_MB] ))as SQLTotalFreeSpaceDB  into #usados  from
#dbfileinfo as A  left join #fixeddrives as B  on substring(A.PhysicalFileName, 1,
1) = B.DriveLetter   group by substring(A.PhysicalFileName, 1, 1)
  select servidor,  DriveLetter,  MB_Free as  RealMb_free,  MB_Free +
SQLTotalFreeSpaceDB as  MB_FreeNeto,  SqlTotalDB,  abs(( SqlTotalDB -
SQLTotalFreeSpaceDB )) as  SQLTotalUsedSpaceDB,  SQLTotalFreeSpaceDB, ( 100 * abs((
SqlTotalDB - SQLTotalFreeSpaceDB )) ) / SqlTotalDB as  Porcentaje_Uso_DB
 from #fixeddrives as f  inner join #usados as z  on z.unidad = f.DriveLetter
  if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#dbfileinfo'))  begin  drop table #dbfileinfo  end
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#logsizestats'))  begin  drop table #logsizestats  end
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#datafilestats'))  begin  drop table #datafilestats  end
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#fixeddrives'))  begin  drop table #fixeddrives  end
 if exists (select 1  from tempdb..sysobjects  where [Id] =
object_id('tempdb..#usados'))  begin  drop table #usados  end
  set nocount off
end
```

Stored Procedures Performance

```
SELECT DB_NAME(database_id) DBName,
 OBJECT_NAME(object_id) SPName,
 datediff(second, last_execution_time, getdate()) SecondsAgo,
 last_execution_time LastExecDate,
 CASE WHEN execution_count = 0 THEN '--' ELSE

RIGHT('0'+convert(varchar(5),(total_elapsed_time/(1000000*execution_count))/3600),2
)+':'+

RIGHT('0'+convert(varchar(5),(total_elapsed_time/(1000000*execution_count))%3600/60
),2)+':'+

RIGHT('0'+convert(varchar(5),((total_elapsed_time/(1000000*execution_count))%60)),2
) END ReadableTime,
 CASE WHEN execution_count= 0 THEN 0 ELSE total_elapsed_time/(1000*execution_count)
END AvgTimeMS,
 CASE WHEN execution_count= 0 THEN 0 ELSE total_worker_time/(1000*execution_count)
END AvgTimeCPU,
 last_elapsed_time/1000 LastTimeMS,
 min_elapsed_time/1000 MinTimeMS,
 total_elapsed_time/1000 TotalTimeMS,
 CASE WHEN DATEDIFF(second, s.cached_time, GETDATE()) < 1 THEN 0 ELSE
 cast(execution_count as decimal) / cast(DATEDIFF(second, s.cached_time, GETDATE())
as decimal) END ExecPerSecond,
 execution_count TotalExecCount,
 last_worker_time/1000 LastWorkerCPU,
 last_physical_reads LastPReads,
 max_physical_reads MaxPReads,
 last_logical_writes LastLWrites,
 last_logical_reads LastLReads
FROM sys.dm_exec_procedure_stats s
WHERE database_id = DB_ID()
AND last_execution_time > dateadd(day, -7, getdate())
ORDER BY 6 desc, 3
```

Check Database Backup file if it's Valid on multiple databases

```
declare @recordTeble table (dataid int identity(1,1), dbname
varchar(100),physicalDevice nvarchar(200))
declare @a int, @z int, @bakfile nvarchar(200), @sql nvarchar(max), @n varchar(2)
insert into @recordTeble
select distinct b.name, bmf.physical_device_name from msdb.dbo.backupset b
JOIN msdb.dbo.backupmediafamily bmf
ON b.media_set_id = bmf.media_set_id
where b.backup_finish_date >= getdate()-1
and bmf.physical_device_name not like'%.trn' and bmf.physical_device_name not
like'%Data Protector%'
select @a=1,@z=MAX(dataid) from @recordTeble
while @a< = @z
begin
select @bakfile = physicalDevice from @recordTeble
where dataid=@a
restore verifyonly from disk = @bakfile
select @a=@a+1
end
```

Check DBCC for all databases

```
declare @db varchar(100),
 @dbid int,
 @hidb int

Select @hidb = Max(dbid ),
 @dbid = 0
from master..sysdatabases

While @dbid <= @hidb
Begin
 Set @db = null
 Select @db = name
 From sysdatabases
 Where dbid = @dbid

 if @db is not null
  DBCC CheckDB( @db )

 Set @dbid = @dbid + 1
End

--Alternate way:
```

```
EXEC sp_msforeachdb 'DBCC CHECKDB(''?'')'
```

Last DBCC checked

```
CREATE TABLE #temp (
     Id INT IDENTITY(1,1),
     ParentObject VARCHAR(255),
     [Object] VARCHAR(255),
     Field VARCHAR(255),
     [Value] VARCHAR(255)
                              )
INSERT INTO #temp
EXECUTE SP_MSFOREACHDB'DBCC DBINFO ( ''?'') WITH TABLERESULTS';

;WITH CHECKDB1 AS
(
    SELECT [Value],ROW_NUMBER() OVER (ORDER BY ID) AS rn1 FROM #temp WHERE Field IN
('dbi_dbname'))
    ,CHECKDB2 AS ( SELECT [Value], ROW_NUMBER() OVER (ORDER BY ID) AS rn2 FROM
#temp WHERE Field IN ('dbi_dbccLastKnownGood')
)
SELECT CHECKDB1.Value AS DatabaseName
        , CHECKDB2.Value AS LastRanDBCCCHECKDB
FROM CHECKDB1 JOIN CHECKDB2
ON rn1 =rn2

DROP TABLE #temp
```

Database last updated

```
Use master

SELECT DatabaseName, MAX(LastAccessDate) LastUpdateDate
FROM
(SELECT
DB_NAME(database_id) DatabaseName
, last_user_update
FROM sys.dm_db_index_usage_stats) AS PivotTable
UNPIVOT
(LastAccessDate FOR last_user_access IN
(last_user_update)
) AS UnpivotTable
GROUP BY DatabaseName
HAVING DatabaseName NOT IN ('master', 'tempdb', 'model', 'msdb')ORDER BY 2
```

Database name from Database ID

```
select DB_Name (83)
```

Instance Restarted

```
SELECT
    login_time
FROM
```

```
    sys.dm_exec_sessions
WHERE
    session_id = 1
```

Operators With Jobs and Notifications

```
-------------------------------------------------------
-- Sysoperators With Jobs
-------------------------------------------------------
--This script will list the operators that have jobs
--that are configured to notify them with alerts.
--The output indicates if the operator is enabled
--or not and also if the job is enabled.
-------------------------------------------------------
SET NOCOUNT ON

SELECT o.[id]
 ,o.[name] AS Operator
 ,CASE WHEN o.[enabled] =1 THEN 'YES' ELSE 'NO' END AS OperatorEnabled
 ,o.[email_address]
 ,o.[last_email_date]
 ,o.[last_email_time]
 ,CASE WHEN j.[enabled] =1 THEN 'YES' ELSE 'NO' END AS JobEnabled
 ,j.[name] AS JobName
 ,j.[description] AS JobDescription
FROM
[msdb].[dbo].[sysoperators] o INNER JOIN
[msdb].[dbo].[sysjobs_view] j ON o.[id] = j.[notify_email_operator_id]
WHERE
j.[notify_email_operator_id] != 0
```