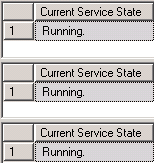
Item 1 - Are all of your SQL Server services running?

Obviously once you connect to your instance you know that the database service is up and running but you can use the [extended stored procedure xp\_servicecontrol](https://www.mssqltips.com/sqlservertip/2036/monitor-start-and-stop-sql-server-services-using-xpservicecontrol/) to check if any service is up and running. Below are a few examples of what you can check.

exec master.dbo.xp\_servicecontrol 'QUERYSTATE', 'MSSQLServer'

exec master.dbo.xp\_servicecontrol 'QUERYSTATE', 'SQLServerAgent'

exec master.dbo.xp\_servicecontrol 'QUERYSTATE', 'SQLBrowser'



*Related tips:*

* [Monitor, Start and Stop SQL Server services using xp\_servicecontrol](https://www.mssqltips.com/sqlservertip/2036/monitor-start-and-stop-sql-server-services-using-xpservicecontrol/)

Item 2 - Did all of your SQL Agent Jobs run successfully?

This item can be checked with a fairly straightforward query of the [msdb database](https://www.mssqltips.com/sqlservertip/1420/sql-server-system-databases/). The first part of the query checks for any [failed job steps](https://www.mssqltips.com/sqlservertip/1054/failed-sql-server-agent-jobs/) and the second part is only concerned with the overall job status. This is also checked because a step could be set to continue even on failure, but should probably still be looked at in the morning. Also, if you are using the SQL Server Agent to [backup your databases](https://www.mssqltips.com/sqlservertip/1251/do-you-know-if-your-sql-server-database-backups-are-successful/) then this is also a good way to check if any backup jobs failed.

use msdb

go

select 'FAILED' as Status, cast(sj.name as varchar(100)) as "Job Name",

cast(sjs.step\_id as varchar(5)) as "Step ID",

cast(sjs.step\_name as varchar(30)) as "Step Name",

cast(REPLACE(CONVERT(varchar,convert(datetime,convert(varchar,sjh.run\_date)),102),'.','-')+' '+SUBSTRING(RIGHT('000000'+CONVERT(varchar,sjh.run\_time),6),1,2)+':'+SUBSTRING(RIGHT('000000'+CONVERT(varchar,sjh.run\_time),6),3,2)+':'+SUBSTRING(RIGHT('000000'+CONVERT(varchar,sjh.run\_time),6),5,2) as varchar(30)) 'Start Date Time',

sjh.message as "Message"

from sysjobs sj

join sysjobsteps sjs

on sj.job\_id = sjs.job\_id

join sysjobhistory sjh

on sj.job\_id = sjh.job\_id and sjs.step\_id = sjh.step\_id

where sjh.run\_status <> 1

and cast(sjh.run\_date as float)\*1000000+sjh.run\_time >

cast(convert(varchar(8), getdate()-1, 112) as float)\*1000000+70000 --yesterday at 7am

union

select 'FAILED',cast(sj.name as varchar(100)) as "Job Name",

'MAIN' as "Step ID",

'MAIN' as "Step Name",

cast(REPLACE(CONVERT(varchar,convert(datetime,convert(varchar,sjh.run\_date)),102),'.','-')+' '+SUBSTRING(RIGHT('000000'+CONVERT(varchar,sjh.run\_time),6),1,2)+':'+SUBSTRING(RIGHT('000000'+CONVERT(varchar,sjh.run\_time),6),3,2)+':'+SUBSTRING(RIGHT('000000'+CONVERT(varchar,sjh.run\_time),6),5,2) as varchar(30)) 'Start Date Time',

sjh.message as "Message"

from sysjobs sj

join sysjobhistory sjh

on sj.job\_id = sjh.job\_id

where sjh.run\_status <> 1 and sjh.step\_id=0

and cast(sjh.run\_date as float)\*1000000+sjh.run\_time >

cast(convert(varchar(8), getdate()-1, 112) as float)\*1000000+70000 --yesterday at 7am

Verify all of your SQL Server Agent Jobs have run successfully

*Related tips:*

* [SQL Server System Databases](https://www.mssqltips.com/sqlservertip/1420/sql-server-system-databases/)
* [Failed SQL Server Agent Jobs](https://www.mssqltips.com/sqlservertip/1054/failed-sql-server-agent-jobs/)
* [Do you know if your SQL Server database backups are successful](https://www.mssqltips.com/sqlservertip/1251/do-you-know-if-your-sql-server-database-backups-are-successful/)
* [SQL Server Agent Tips](https://www.mssqltips.com/sql-server-tip-category/27/sql-server-agent/)

Item 3 - Do you have a recent backup of all your SQL Server databases?

The two queries below will list any database that either does not have any backup or [has not been backed up in the last 24 hours](https://www.mssqltips.com/sqlservertip/1251/do-you-know-if-your-sql-server-database-backups-are-successful/). The first query checks your full backups and the second query checks your transaction log backups (only for those databases in full recovery mode).

SELECT d.name AS "Database",

ISNULL(CONVERT(VARCHAR,b.backupdate,120),'NEVER') AS "Last Full Backup"

FROM sys.databases d

LEFT JOIN (SELECT database\_name,type,MAX(backup\_finish\_date) backupdate FROM backupset

WHERE type LIKE 'D'

GROUP BY database\_name,type) b on d.name=b.database\_name

WHERE (backupdate IS NULL OR backupdate < getdate()-1)

SELECT d.name AS "Database",

ISNULL(CONVERT(VARCHAR,b.backupdate,120),'NEVER') AS "Last Log Backup"

FROM sys.databases d

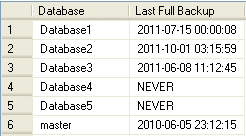
LEFT JOIN (SELECT database\_name,type,MAX(backup\_finish\_date) backupdate FROM backupset

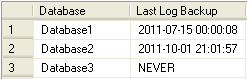
WHERE type LIKE 'L'

GROUP BY database\_name,type) b on d.name=b.database\_name

WHERE recovery\_model = 1

AND (backupdate IS NULL OR backupdate < getdate()-1)





*Related tips:*

* [Do you know if your SQL Server database backups are successful](https://www.mssqltips.com/sqlservertip/1251/do-you-know-if-your-sql-server-database-backups-are-successful/)
* [SQL Server Backup and Recovery Tips](https://www.mssqltips.com/sql-server-tip-category/8/backup-and-recovery/)

Item 4 - Are there any errors in your SQL Server Error Log?

In order to check the [SQL Server Error Log](https://www.mssqltips.com/sqlservertip/1476/reading-the-sql-server-log-files-using-tsql/) we are going to use the undocumented extended stored procedure, xp\_readerrorlog. This query will look at the current log and go back a maximum of 2 days looking for any errors during that time frame.

declare @Time\_Start datetime;

declare @Time\_End datetime;

set @Time\_Start=getdate()-2;

set @Time\_End=getdate();

-- Create the temporary table

CREATE TABLE #ErrorLog (logdate datetime

, processinfo varchar(255)

, Message varchar(500))

-- Populate the temporary table

INSERT #ErrorLog (logdate, processinfo, Message)

EXEC master.dbo.xp\_readerrorlog 0, 1, null, null , @Time\_Start, @Time\_End, N'desc';

-- Filter the temporary table

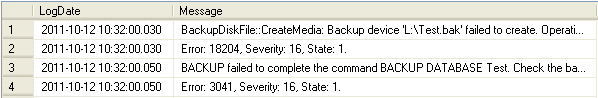
SELECT LogDate, Message FROM #ErrorLog

WHERE (Message LIKE '%error%' OR Message LIKE '%failed%') AND processinfo NOT LIKE 'logon'

ORDER BY logdate DESC

-- Drop the temporary table

DROP TABLE #ErrorLog



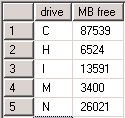
*Related tips:*

* [Reading the SQL Server log files using TSQL](https://www.mssqltips.com/sqlservertip/1476/reading-the-sql-server-log-files-using-tsql/)
* [Read the end of a large SQL Server Error Log](https://www.mssqltips.com/sqlservertip/1866/read-the-end-of-a-large-sql-server-error-log/)

Item 5 - Are you running out of space on any of your disks on your SQL Server?

You can used the extended stored procedure xp\_fixeddrives to get a quick look at the space left on your drives.

exec master.dbo.xp\_fixeddrives



*Related tips:*

* [SQLCLR function to return free space for all drives on a server](https://www.mssqltips.com/sqlservertip/1986/sqlclr-function-to-return-free-space-for-all-drives-on-a-server/)
* [Determine Free Disk Space in SQL Server with TSQL Code](https://www.mssqltips.com/sqlservertip/1706/determine-free-disk-space-in-sql-server-with-tsql-code/)
* [Script to Get Available and Free Disk Space for SQL Server](https://www.mssqltips.com/sqlservertip/2444/script-to-get-available-and-free-disk-space-for-sql-server/)

Item 6 - Are you running low on server memory for SQL Server?

To check the memory on your server we can use the dynamic management view dm\_os\_sys\_memory.

SELECT available\_physical\_memory\_kb/1024 as "Total Memory MB",

available\_physical\_memory\_kb/(total\_physical\_memory\_kb\*1.0)\*100 AS "% Memory Free"

FROM sys.dm\_os\_sys\_memory

SQL Server Memory Usage

*Related tips:*

* [How to Identify Microsoft SQL Server Memory Bottlenecks](https://www.mssqltips.com/sqlservertip/2304/how-to-identify-microsoft-sql-server-memory-bottlenecks/)
* [Dynamic Management Views and Functions](https://www.mssqltips.com/sql-server-tip-category/31/dynamic-management-views-and-functions/)

Item 7 - Are there any SQL Server statements in the cache that could use tuning?

The following query will identify any poor performing SQL statements. You can alter the "order by" clause depending on what you are most concerned with (IO vs. CPU vs. Elapsed Time).

SELECT top 10 text as "SQL Statement",

last\_execution\_time as "Last Execution Time",

(total\_logical\_reads+total\_physical\_reads+total\_logical\_writes)/execution\_count as [Average IO],

(total\_worker\_time/execution\_count)/1000000.0 as [Average CPU Time (sec)],

(total\_elapsed\_time/execution\_count)/1000000.0 as [Average Elapsed Time (sec)],

execution\_count as "Execution Count",

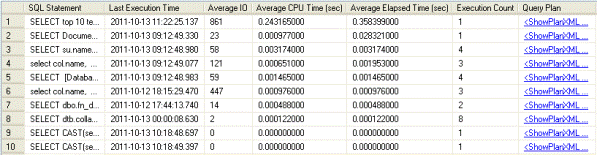
qp.query\_plan as "Query Plan"

FROM sys.dm\_exec\_query\_stats qs

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

CROSS APPLY sys.dm\_exec\_query\_plan(qs.plan\_handle) qp

order by total\_elapsed\_time/execution\_count desc



*Related tips:*

* [SQL Server Graphical Query Plan Tutorial](https://www.mssqltips.com/sqlservertutorial/2250/graphical-query-plan-tutorial/)
* [Dynamic Management Views and Functions](https://www.mssqltips.com/sql-server-tip-category/31/dynamic-management-views-and-functions/)

Item 8 - How many connections do you have to your SQL Server instance?

This query on its own does not provide too much information other than show you if there is some blocking in the system. However, once you get a baseline for your applications through running this query, you'll be able to see if you have a higher than normal number of connections. This can be an early sign that there may be a problem.

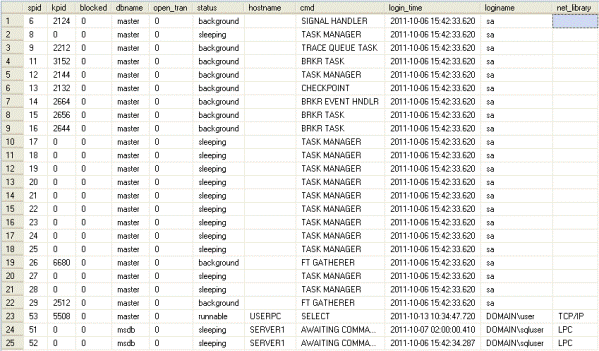
SELECT spid, kpid, blocked, d.name, open\_tran, status, hostname,

cmd, login\_time, loginame, net\_library

FROM sys.sysprocesses p

INNER JOIN sys.databases d

on p.dbid=d.database\_id



Item 9 - How many requests is your SQL Server processing?

As with the previous query, checking the number of requests coming into your SQL Server does not tell you too much. If you capture this number during normal operation you can use it as a baseline for comparison later on. Generally speaking around 1000/sec is a busy SQL Server but this number depends a lot on the hardware you are running on. For others 100/sec may be too much for their instance to handle. Also, using the below query as a template, you can query other O/S performance counters related to SQL Server.

DECLARE @BRPS BIGINT

SELECT @BRPS=cntr\_value

FROM sys.dm\_os\_performance\_counters

WHERE counter\_name LIKE 'Batch Requests/sec%'

WAITFOR DELAY '000:00:10'

SELECT (cntr\_value-@BRPS)/10.0 AS "Batch Requests/sec"

FROM sys.dm\_os\_performance\_counters

WHERE counter\_name LIKE 'Batch Requests/sec%'

SQL Server Batch Requests Per Second

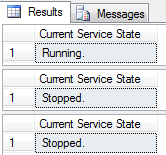
## SQL Server Service Health Check

The first thing you need to ensure is that all your SQL Server services are alive. Of course, if you are able to successfully connect to SQL Server instance it means that it is alive, but still, you can check the status of all your SQL Server services using the following commands:

exec master.dbo.xp\_servicecontrol 'QUERYSTATE', 'MSSQLServer'

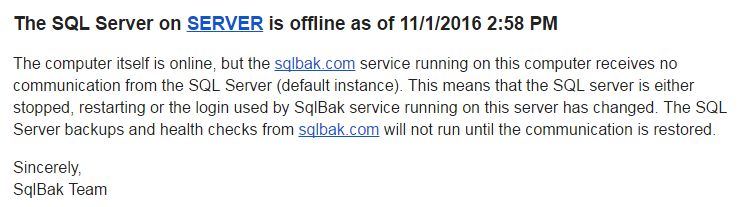
exec master.dbo.xp\_servicecontrol 'QUERYSTATE', 'SQLServerAgent'

exec master.dbo.xp\_servicecontrol 'QUERYSTATE', 'SQLBrowser'

Here you can check which services are running and which stopped:

Of course, it’s a bit weird to run this query manually every hour, isn’t it? As a developer, you should probably think about creating some simple application or a batch file that will check the status of your SQL Server and notify you if something goes wrong.

[SqlBak](https://sqlbak.com/) users don’t need any additional application though. It performs regular checks and sends alerts in the event of service failure.



## SQL Server Backup Health Check

Another thing you need to check is database backups. Again, you can do it either manually or with [SqlBak](https://sqlbak.com/" \t "_blank).

To check it manually you can use the following two useful queries. The first one checks your full backups and the second one checks all [transaction log](http://sqlbak.com/academy/transaction-log/)backups:

SELECT d.name AS "Database",

ISNULL(CONVERT(VARCHAR,b.backupdate,120),'NEVER') AS "Last [Full Backup](http://sqlbak.com/academy/full-backup/)"

FROM sys.databases d

LEFT JOIN (SELECT database\_name,type,MAX(backup\_finish\_date) backupdate FROM backupset

WHERE type LIKE 'D'

GROUP BY database\_name,type) b on d.name=b.database\_name

WHERE (backupdate IS NULL OR backupdate < getdate()-1)

SELECT d.name AS "Database",

ISNULL(CONVERT(VARCHAR,b.backupdate,120),'NEVER') AS "Last Log Backup"

FROM sys.databases d

LEFT JOIN (SELECT database\_name,type,MAX(backup\_finish\_date) backupdate FROM backupset

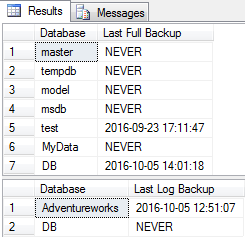
WHERE type LIKE 'L'

GROUP BY database\_name,type) b on d.name=b.database\_name

WHERE recovery\_model = 1

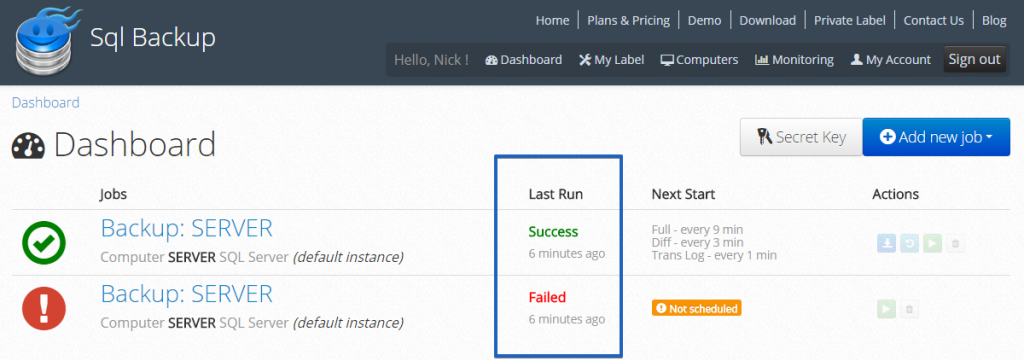
AND (backupdate IS NULL OR backupdate < getdate()-1)

As a result you will see a list of databases that either do not have any backups at all or have not been backed up in the past 24 hours:

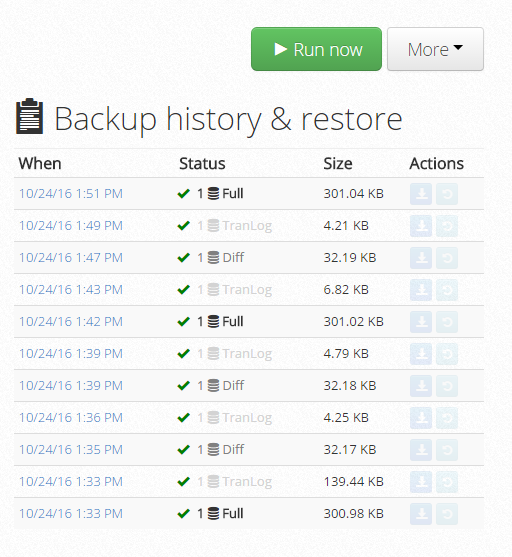


However if you use [SqlBak](https://sqlbak.com/" \t "_blank) to backup your databases, you don’t need to check your SQL Server database backups manually all the time. You will be notified automatically if any or your scheduled backups failed.

At the same time, if you want to see the status of all backup jobs you can open your [Dashboard](https://sqlbak.com/app/dashboard) page and check the “Last Run” column. If the last backup succeeded there will be “Success” sign and if it failed, then “Failed”.



In addition, you can check the entire database backup history by clicking on the backup job you need:



## Check for Available Disk Space

One of the trickiest problems DBA may face is insufficient disk space. If you run out of free disk space it will not only slow your server down but also may cause multiple errors and application failures. And the most dangerous thing here is that it may happen when you least expect it (for example in the midst of your vacation).

You can check free disk space using a simple Windows command:

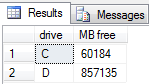
fsutil volume diskfree c:

or with a bit more complicated WMI command:

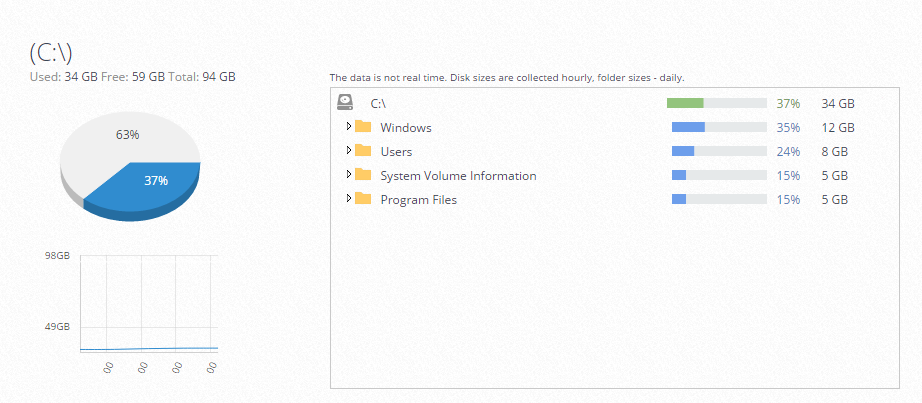
wmic /node:"%COMPUTERNAME%" LogicalDisk Where DriveType="3" Get DeviceID,FreeSpace|find /I "c:"

If you need you can check how much free disk space is left using the following SQL Server statement:

exec master.dbo.xp\_fixeddrives



But why not let [SqlBak](https://sqlbak.com/" \t "_blank) do it for you automatically? It has a nice feature that will perform hourly checks and alert you via email when free disk space crosses the threshold you set.

Additionally, SqlBak’s Health Check dashboard allows you to easily determine what folders and files occupy your disk:

## Check for free memory available for SQL Server

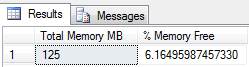
Besides monitoring server’s free disk space you also need to take note of how much free memory is available on your server. If the amount of free memory is approaching some critical value you need to take steps to free it.

To check your server’s memory with SQL you can use the following syntax:

SELECT available\_physical\_memory\_kb/1024 as "Total Memory MB",

available\_physical\_memory\_kb/(total\_physical\_memory\_kb\*1.0)\*100 AS "% Memory Free"

FROM sys.dm\_os\_sys\_memory

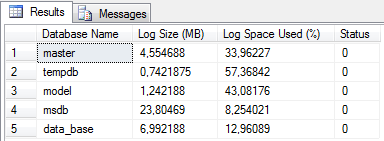


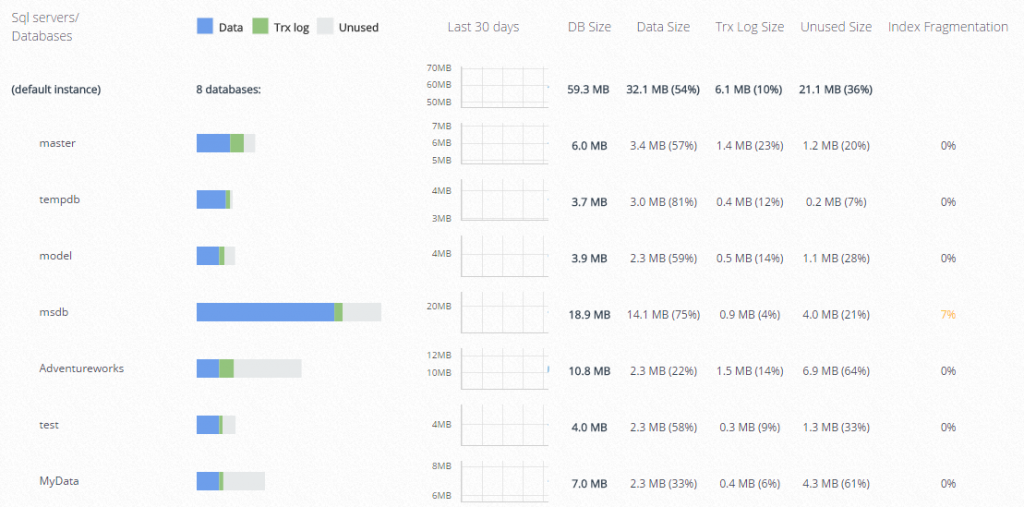
## Check the size of the transaction log

The [transaction log](http://sqlbak.com/academy/transaction-log/)is one of the crucial aspects of every database. It is used to keep all transactions prior to committing the data into the data file. Depending on the number of transactions it may happen that transaction logs grow quite large and not knowing what is in the transaction log or how much space is occupied by it can cause some problems.

To get the size of transaction logs you can run the following query:

DBCC SQLPERF(LOGSPACE)



[SqlBak](https://sqlbak.com/) Automatic Health Check can help you here as well as it monitors how much space does the transaction log occupy for each database:

## Check for Index Fragmentation

Besides fragmentation at the file system level it’s also possible to have fragmentation within the data and log files, in the structures that store the log, table and index data. This also may lead to a significant drop in database performance.

You can use the system function [sys.dm\_db\_index\_physical\_stats](https://msdn.microsoft.com/en-us/library/ms188917.aspx" \t "_blank) to detect fragmentation in data and indexes of the specified table or view in SQL Server. Run the following query

SELECT OBJECT\_NAME(OBJECT\_ID), index\_id,index\_type\_desc,index\_level,

avg\_fragmentation\_in\_percent,avg\_page\_space\_used\_in\_percent,page\_count

FROM sys.dm\_db\_index\_physical\_stats

(DB\_ID(N'<YOUR DATABASE>'), NULL, NULL, NULL , 'SAMPLED')

ORDER BY avg\_fragmentation\_in\_percent DESC

and you will get basic information about external fragmentation percentage and an average percentage of use of pages that represents the internal fragmentation.

And as shown in the screenshot above, SqlBak’s SQL Server Health Check also provides information about index fragmentation.