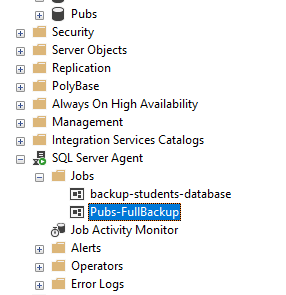
# Week 13 Practical

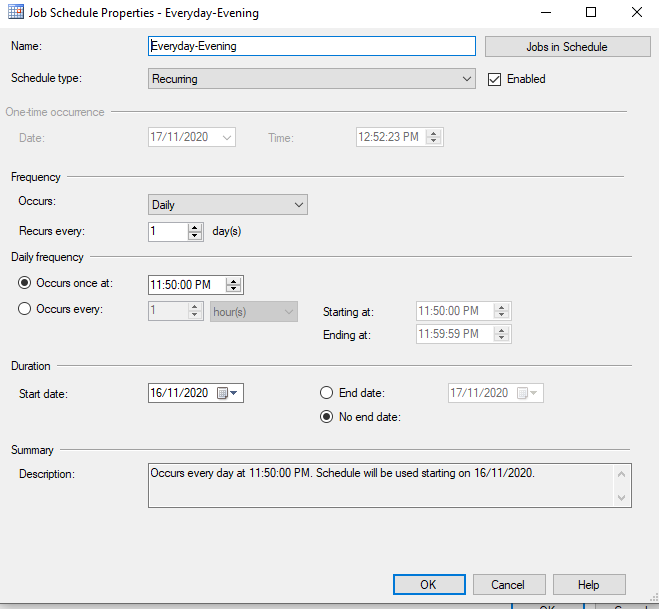
# Duties of a DBA 10%

## Examine each of these carefully, screen captures and notes can be used as proof of completion. Perform any additional setup as required to complete these tasks.

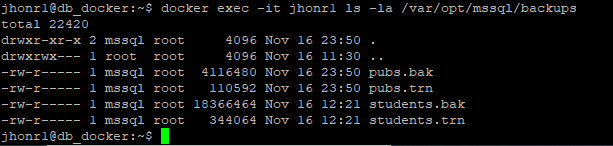
* Check the previous night’s SQL Server database and transaction log backups and SQL Server Agent jobs for errors.

**SQL Server Agent – Auto backup:**

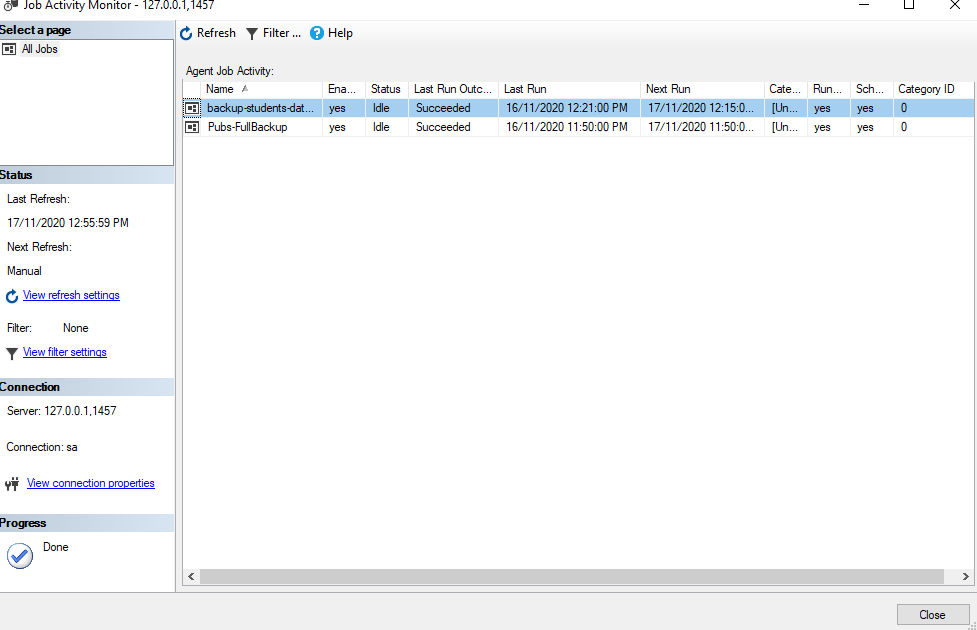




**Auto backup in docker container mssql server:**



**SQL SERVER AGENT Error check:**

****

* Check all databases to make sure all are up and not marked as suspect. Check previous DBCC CHECKDB for errors.

**SCRIPT :**

**To get all databases one by one and run DBCC CHECKDB command.**

DECLARE @maximumRows int

DECLARE @currentRow int

DECLARE @\_DBName nvarchar(100)

DECLARE @ExeTemp nvarchar(1000)

SET @maximumRows = (select Count(\*) from sys.databases)

SET @currentRow = 1

WHILE @currentRow <= @maximumRows

BEGIN

WITH T1 AS

( SELECT

ROW\_NUMBER() OVER(ORDER BY name) AS RowNumber, name

FROM sys.databases

)

SELECT @\_DBName = name

FROM T1

WHERE RowNumber = @currentRow

SET @ExeTemp = 'USE ' + @\_DBName + ' DBCC CHECKDB'

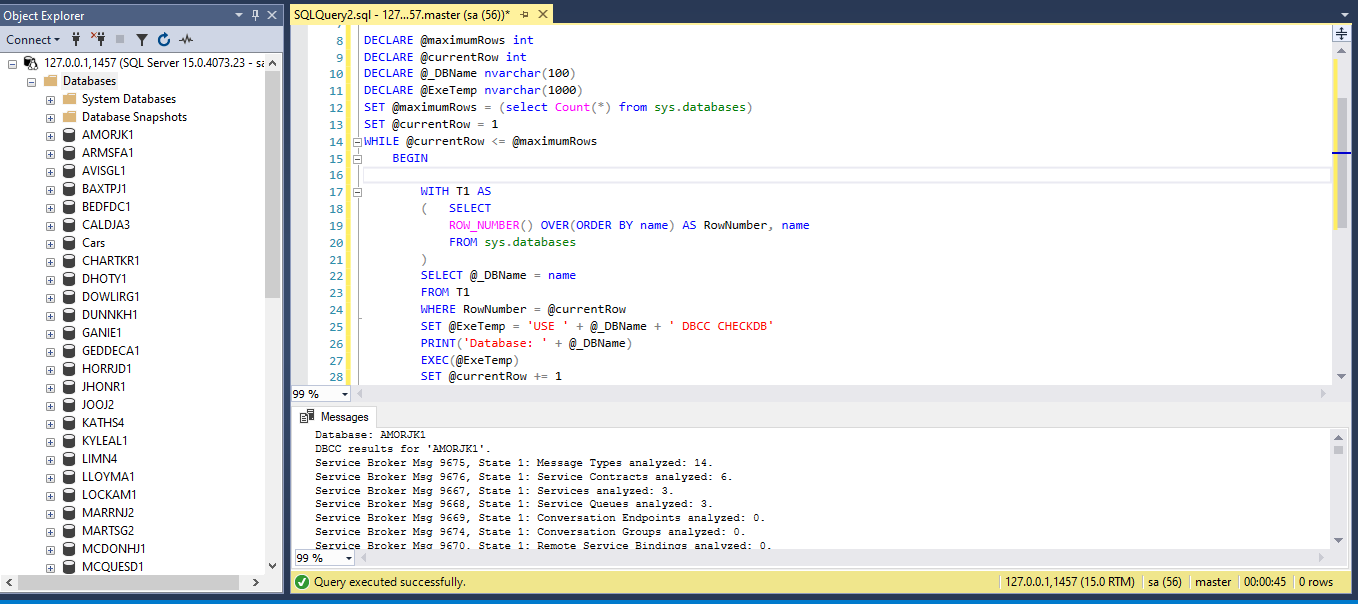
PRINT('Database: ' + @\_DBName)

EXEC(@ExeTemp)

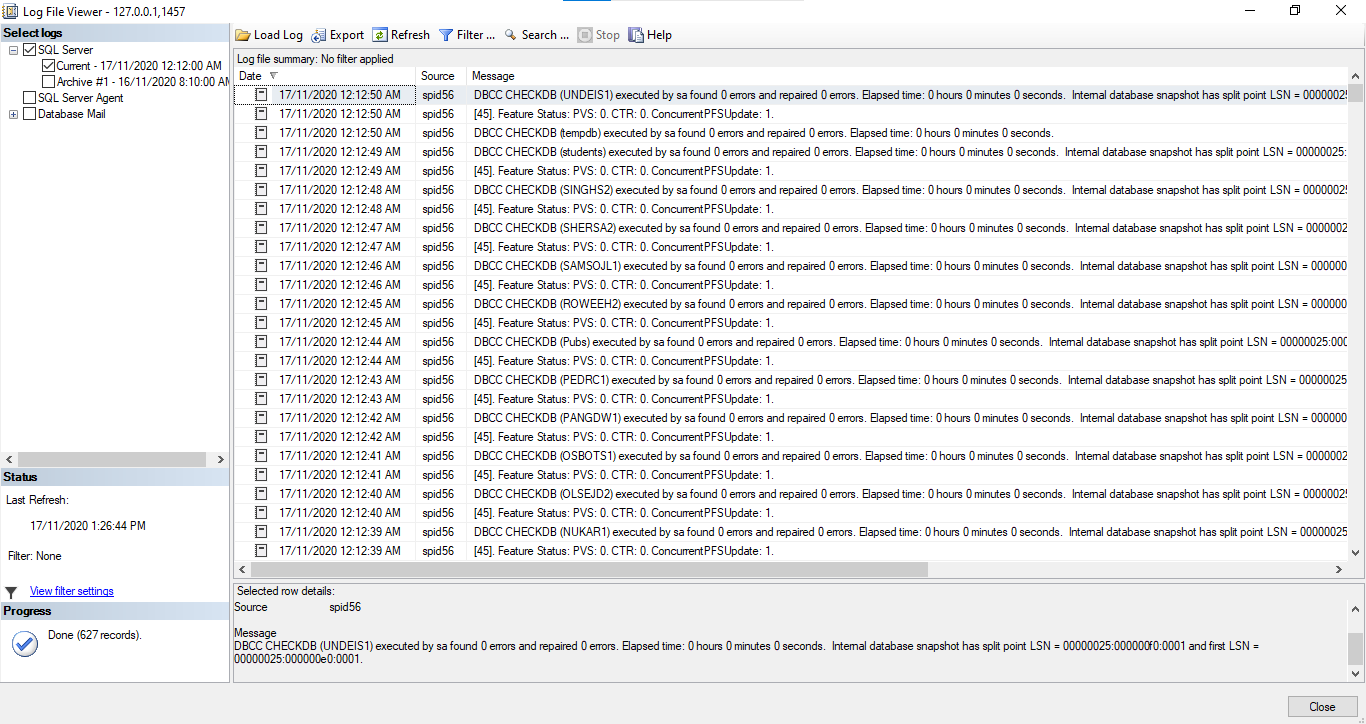
SET @currentRow += 1

END

**Output:**



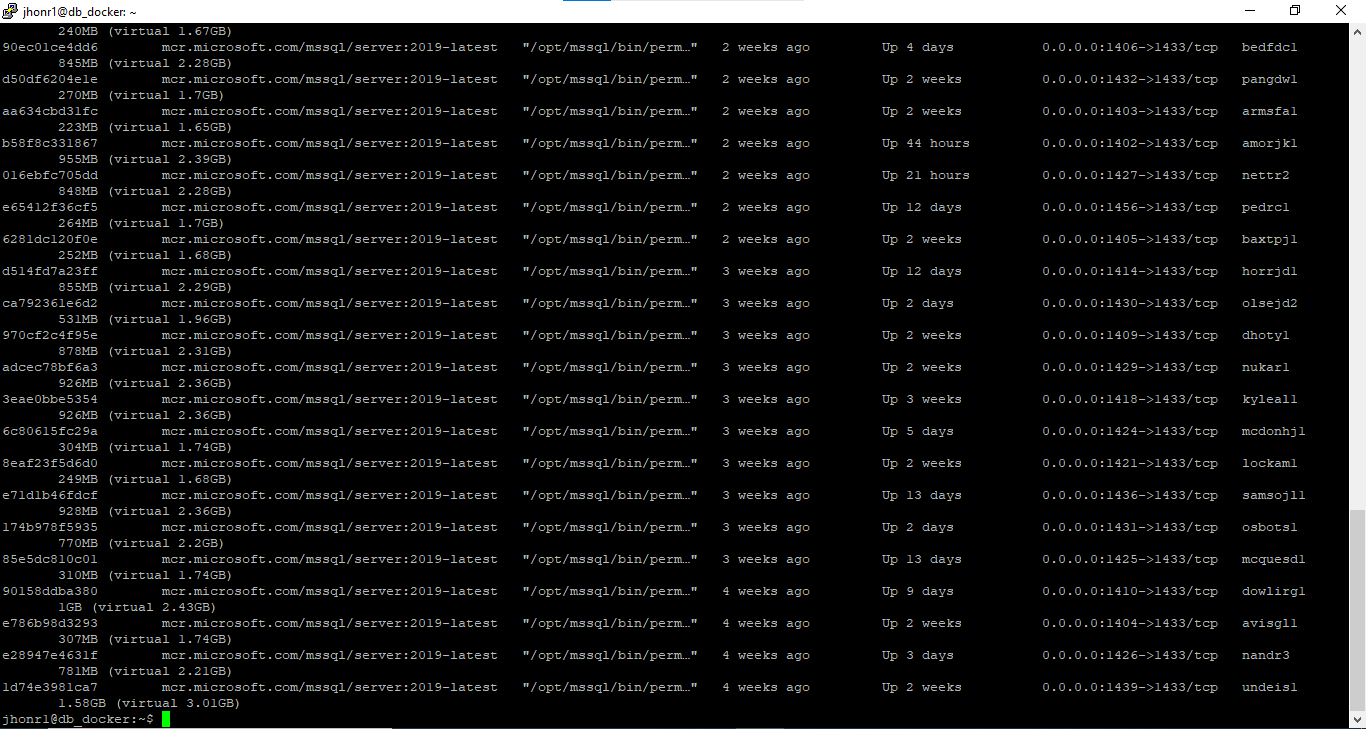
* Check SQL Server Log File entries for warnings and errors and determine if any entries warrant further investigation. Export and save the current log file.



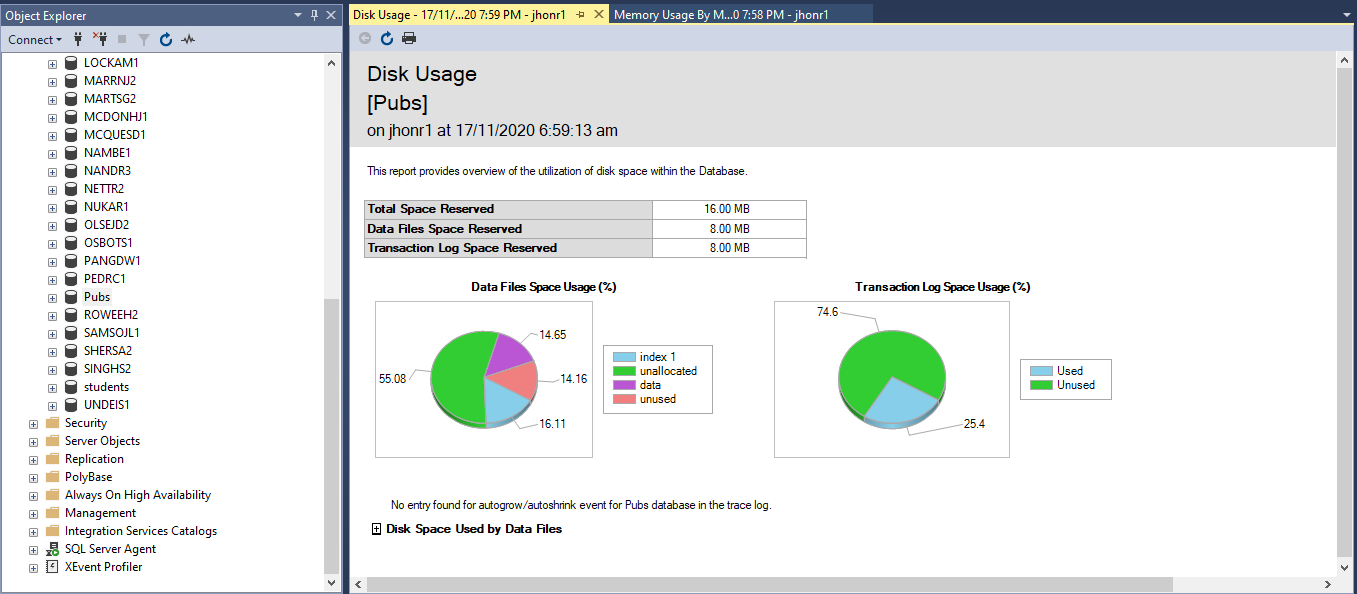
**Log file:**

LogFile-DBCC-CHECKDB.log

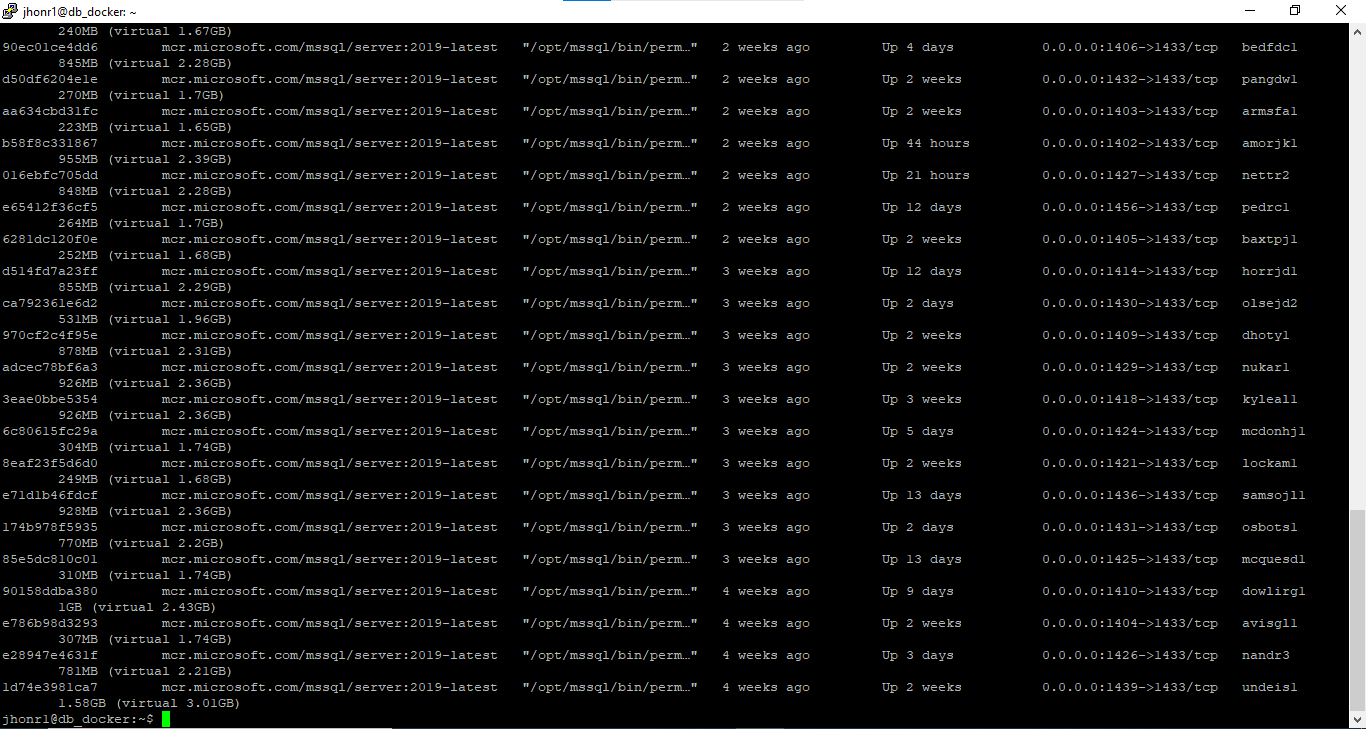
* Look for any security policy violations. Look for resources on the server, such as file sizes and disk space, and audit growth for long-term projections.



* Explore using long-running queries or tasks, Perfmon, etc. to generate data. Set up a sensible logging report to monitor disk and memory usage.



* What about your Container? What sort of usage data can you extract.



### Use your pubs database

DBA task

You want to recover a single table from a database backup – why?

It is the only table effected by a recent data loss.

Restoring an entire backup can take a significant amount of time, and you are under a lot of pressure to get it done fast

Perform the necessary data adjustments to check your solution is correct (delete rows etc). Provide an answer for each of the following scenarios:

1. The table still exists, but only some rows were deleted, restore the deleted data only.

Answer:

1. First I took backup of Pubs database.

BACKUP DATABASE Pubs TO DISK = '/var/opt/mssql/backups/pubs.bak'

1. Now I restored the Pubs database backup “pubs.bak” to a different name “Pubs\_Restored”.

RESTORE DATABASE Pubs\_Restored FROM DISK = '/var/opt/mssql/backups/pubs.bak'

WITH MOVE 'Pubs' TO '/var/opt/mssql/data/pubs\_restored.mdf',

MOVE 'Pubs\_log' TO '/var/opt/mssql/data/pubs\_restored.ldf'

1. Now I had 2 databases Pubs, Pubs\_Restored.
2. Now I needed to delete some rows in anyone of table which has primary key data type as “int”. So, I found that “jobs” table’s “job\_id” has “int” data type.
3. In order to delete some rows in [Pubs].[dbo].[jobs], I needed to delete the foreign key in [Pubs].[dbo].[employee] table. So I deleted some rows in [Pubs].[dbo].[employee] (Eg: job\_id = 4 and 5)

USE Pubs

GO

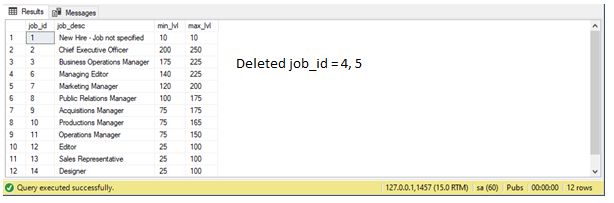
DELETE employee where job\_id = 4

DELETE employee where job\_id = 5

1. Now I can delete rows which has job\_id = 4 and 5. So I deleted rows that matches job\_id = 4, 5.

DELETE jobs where job\_id = 4

DELETE jobs where job\_id = 5



1. Then I executed the script, it checks the table [Pubs].[dbo].[jobs] with table [Pubs\_Restored].[dbo].[jobs].
2. If jobs\_id doesn’t match, it adds new row to [Pubs].[dbo].[jobs] table, else it updates [Pubs].[dbo].[jobs] table.

**SCRIPT:**

USE Pubs

GO

SET IDENTITY\_INSERT jobs ON

MERGE jobs dest

USING (SELECT \* FROM [Pubs\_Restored].[dbo].[jobs] src) AS src

ON dest.job\_id = src.job\_id

WHEN MATCHED THEN UPDATE

SET dest.job\_desc = src.job\_desc, dest.min\_lvl = src.min\_lvl, dest.max\_lvl = src.max\_lvl

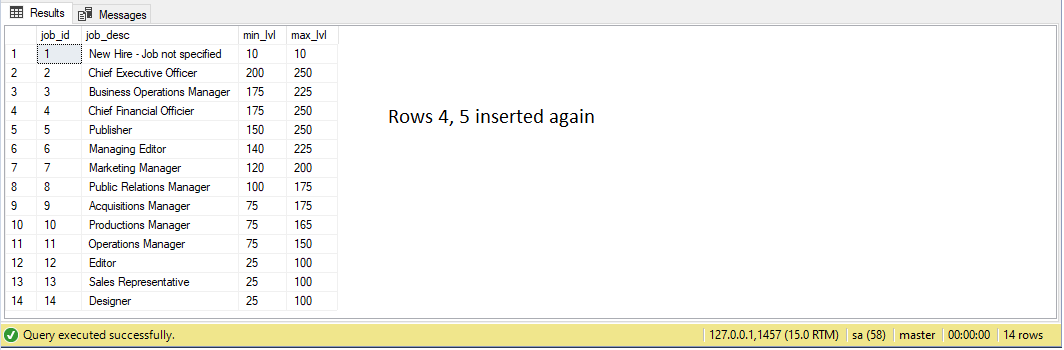
WHEN NOT MATCHED THEN INSERT

(job\_id, job\_desc, min\_lvl, max\_lvl) VALUES

(src.job\_id, src.job\_desc, src.min\_lvl, src.max\_lvl);

SET IDENTITY\_INSERT jobs OFF

**RESULT AFTER RUNNING SCRIPT:**



1. The table has been too badly damaged; restore the table structure and all the data.

Answer:

1. First I took backup of Pubs database.

BACKUP DATABASE Pubs TO DISK = '/var/opt/mssql/backups/pubs.bak'

1. Now I restored the Pubs database backup “pubs.bak” to a different name “Pubs\_Restored”.

RESTORE DATABASE Pubs\_Restored FROM DISK = '/var/opt/mssql/backups/pubs.bak'

WITH MOVE 'Pubs' TO '/var/opt/mssql/data/pubs\_restored.mdf',

MOVE 'Pubs\_log' TO '/var/opt/mssql/data/pubs\_restored.ldf'

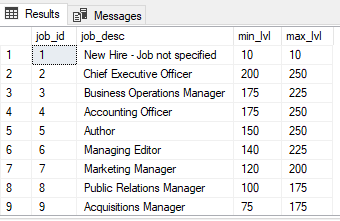
1. Now I had 2 databases Pubs, Pubs\_Restored.
2. I updated some rows in [Pubs].[dbo].[employee] (Eg: job\_id = 4 and 5)

USE Pubs

GO

UPDATE jobs set job\_desc = 'Accounting Officer' where job\_id = 4

UPDATE jobs set job\_desc = 'Author' where job\_id = 5



1. Then I executed the script, it checks the table [Pubs].[dbo].[jobs] with table [Pubs\_Restored].[dbo].[jobs].
2. If jobs\_id matches to [Pubs].[dbo].[jobs] table, it updates [Pubs].[dbo].[jobs] table.

**SCRIPT:**

USE Pubs

GO

SET IDENTITY\_INSERT jobs ON

MERGE jobs dest

USING (SELECT \* FROM [Pubs\_Restored].[dbo].[jobs] src) AS src

ON dest.job\_id = src.job\_id

WHEN MATCHED THEN UPDATE

SET dest.job\_desc = src.job\_desc, dest.min\_lvl = src.min\_lvl, dest.max\_lvl = src.max\_lvl

WHEN NOT MATCHED THEN INSERT

(job\_id, job\_desc, min\_lvl, max\_lvl) VALUES

(src.job\_id, src.job\_desc, src.min\_lvl, src.max\_lvl);

SET IDENTITY\_INSERT jobs OFF

**RESULT AFTER RUNNING SCRIPT:**

