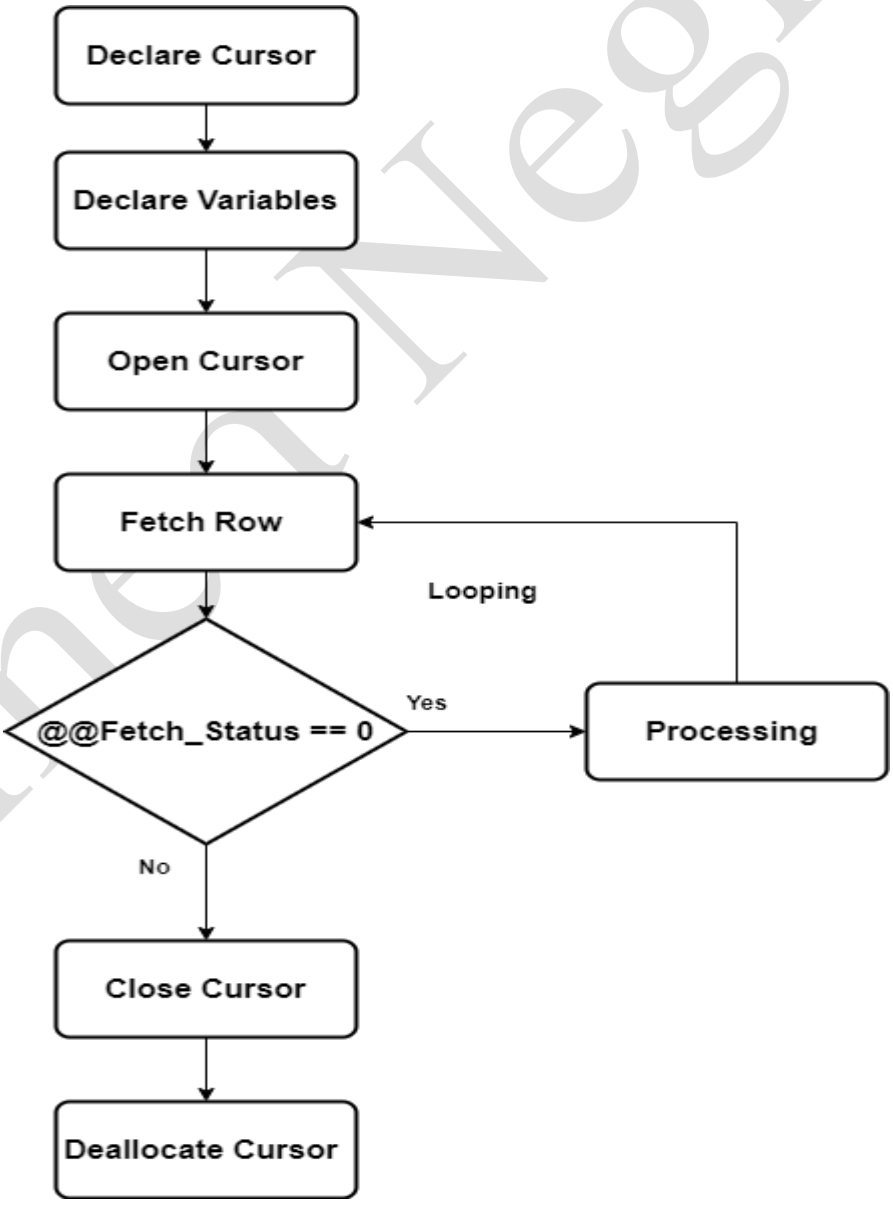
Day 10

*** Cursor :**

- هو Mechanism للتعامل مع الـ Result Set عن طريق Row by Row

- كأنى بـ Loop على الـ Result Set

*** Cursor Life Cycle :**



1) Declare Cursor :

- بحدد الـ Select اللى هنشتغل عليها

- بحدد أنا هـ read only بس من الـ table بدون تعديل على الـ Table أو هعدل فيه من خلال الـ keywords :

1) Read only

2) Update

2) Declare Variables :

- بعمل Variables عشان تـ Hold الـ Columns' Data

3) Open Cursor :

- بـ Place الـ Cursor على الـ first row فى الـ result set

4) Fetch Row :

- بيـ fetch الـ Row اللى Cursor عليه فى الـ Memory

5) Fetch Check :

- بيـ check على الـ Fetch اللى حصلت عن طريق global Variable وهو الـ @@Fetch_status

(بيـ return 3 values

0 : الـ previous fetch is done

1 : معناه إن فيه row موجود بس حصل error فى الـ fetch

2 : معناه إن فيه rows موجودة بس خلصت (مفيش Fetch تانى))

6) Close Cursor :

- بيـ save الـ pointer (Checkpoint)

7) Deallocate Cursor :

- بيـ Remove الـ Memory Space اللى إتحجزت لـ cursor

- الـ Close أقدر أعمل بعدها open عادى (الـ Cursor لسه موجود فى الـ Memory)
- الـ deallocate مقدرش أعمل open (لازم أعمل declare cursor تانى)

Syntax of Cursor :

```
declare Cursor_Name Cursor
for Select Column1_Name ,…… ColumnN_Name
    from Table_Name
    where Condition
for read only \ Update

declare @Var1_Name Datatype ,.....@VarN_Name Datatype
open Cursor_Name
fetch Cursor_Name into @Var1_Name,....@VarN_Name  --counter=0
while  @@FETCH_STATUS=0
    begin
        Select @Var1_Name,.....@VarN_Name--(Processing Query)

        fetch Cursor_Name into @Var1_Name,..@VarN_Name
        -- Counter++
    end
close Cursor_Name
deallocate Cursor_Name
```

- الناتج هيطلع separated

- لو هستخدم الـ Read only Keyword هيبقى معاها فى الـ processing (Select Statemennt)

- لو هستخدم الـ Update Keyword هيبقى معاها فى الـ processing (DML Queries)

(فى الـ Update هستخدم Keyword معاها عشان يحدد الـ row اللى واقف عليه الـ cursor وهى كالأتى : Cursor_Name where current of)

- الـ cursor عامل زى الـ subquery (أخر حل بلجأله)

*** Backup \ Restore :**

- Microsoft Backup types :

1) Full Backup

2) Differential Backup

3) Transaction log Backup

4) File Group Backup

---

1) Full Backup :

- بيتم عمل الـ backup لـ Mdf file (الـ Data)

- من لحظة الـ Database Creation حتى اللحظة اللى بعمل فيها الـ Full Backup

2) Differential Backup :

- بيتم عمل الـ backup لـ Mdf file (الـ Data)

- من أخر Full Backup إتعمل حتى اللحظة اللى بعمل فيها Differential Backup

- لازم أكون عامل Full Backup قبل كدا

- أسرع شوية من الـ Full Backup

3) Transaction Log Backup:

- بيتم عمل الـ Backup لـ Ldf file (الـ Transactions)

- من أخر Backup بغض النظر عن نوعه حتى اللحظة اللى بيتعمل فيها Transaction log backup

- بياخد الـ Queries اللى اتنفذت على الـ Database من أخر Backup و يفضى الـ Log file

4) File group Backup :

- من المعروف إن ممكن أقسم الـ Mdf Files على File groups ودا بيسرع الـ processing

(فممكن أعمل backup لـ file groups اللى إتعدلت بس)

- الـ Full والـ Differential لازم أعمل Restore لـ Backup ككل (مش هعرف أخد جزء منه)

- الـ Transaction log Backup (أقدر أخد جزء من الـ queries اللى اتعملت لحد وقت معين لأن كل query متسجل معاها الوقت اللى اتنفذت فيه)

الطريقة الصحيحة لأنى أعمل Backup :

- كل فترة زمنية بعيدة (شهر مثلا) أعمل Full Backup

- كل فترة زمنية أقصر شوية (أسبوع) أعمل Differential Backup

- كل فترة زمنية أقل (يوم \ ساعة) أعمل Transaction log Backup (سريع مش بيوقف الـ DB)

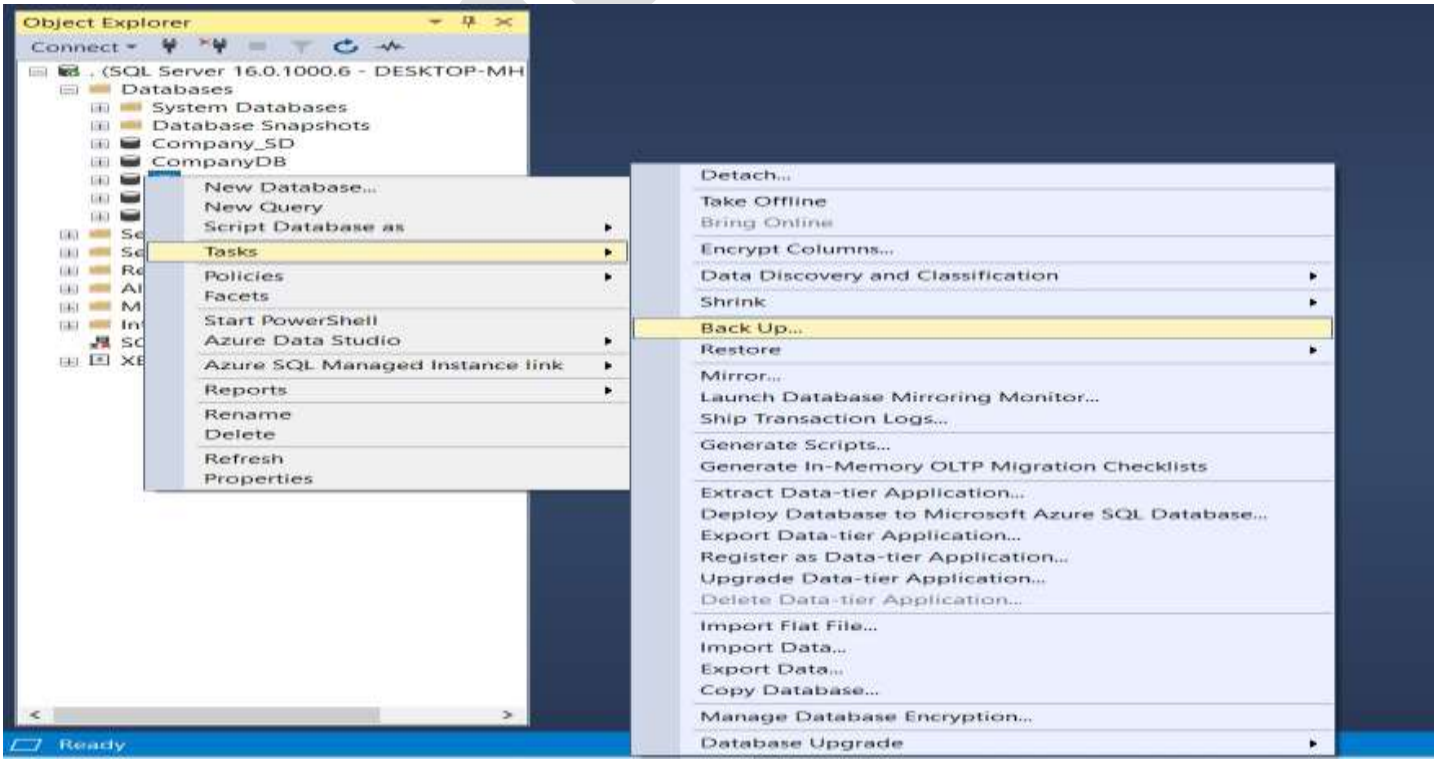
طريقة عمل الـ backup :

1) عن طريق الـ Wizard

2) عن طريق الـ Query

*** Backup Via Wizard :**

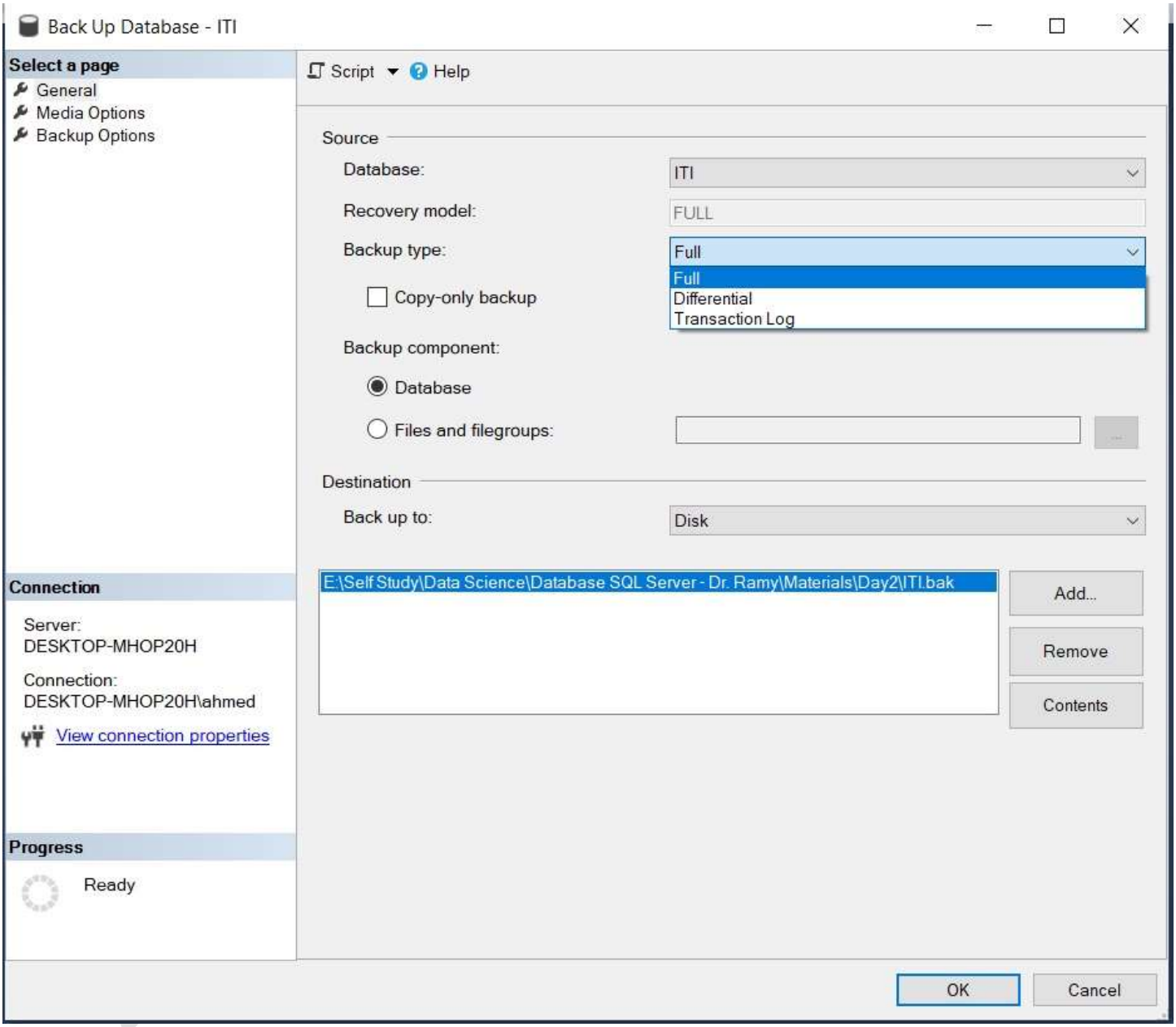
DB Right Click → Tasks → Backup



- هيظهر الـ Backup Wizard وتقدر من خلالها تحدد :

1) إسم الـ bakup والـ Path

2) نوع الـ Backup



*** Backup via Query :**

- ممكن أعمل Backup عن طريق الـ query :

1) FULL :

```
backup database Database_Name
to disk = 'Path'
```

2) Differential :

```
backup database Database_Name
to disk = 'Path'

with differential
```

3) Transactional :

```
backup log Database_Name
to disk = 'Path'
```

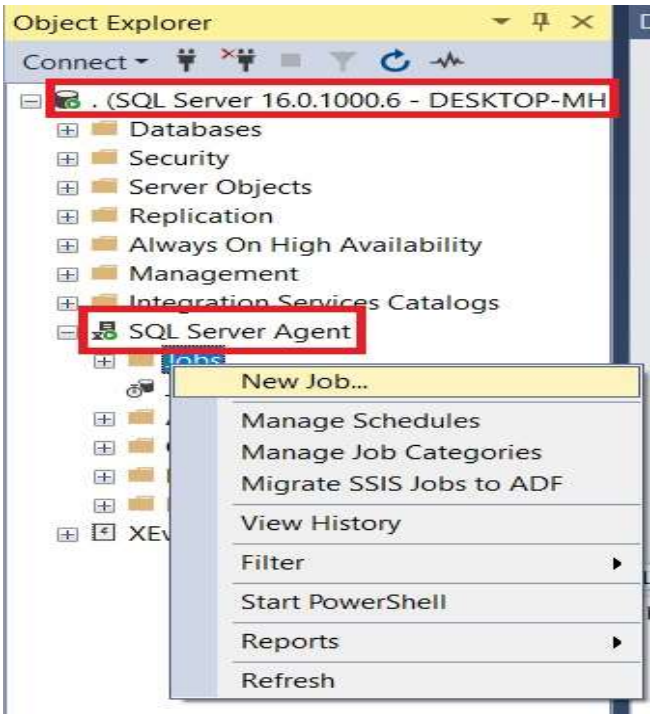
- تـ Run على الـ Master DB

- كدا فى الـ Run هيعمل Backup طب لو عاوز الموضوع يتم بوقت معين (يوم كذا الساعة كذا تعمل Backup) هستخدم الـ Job

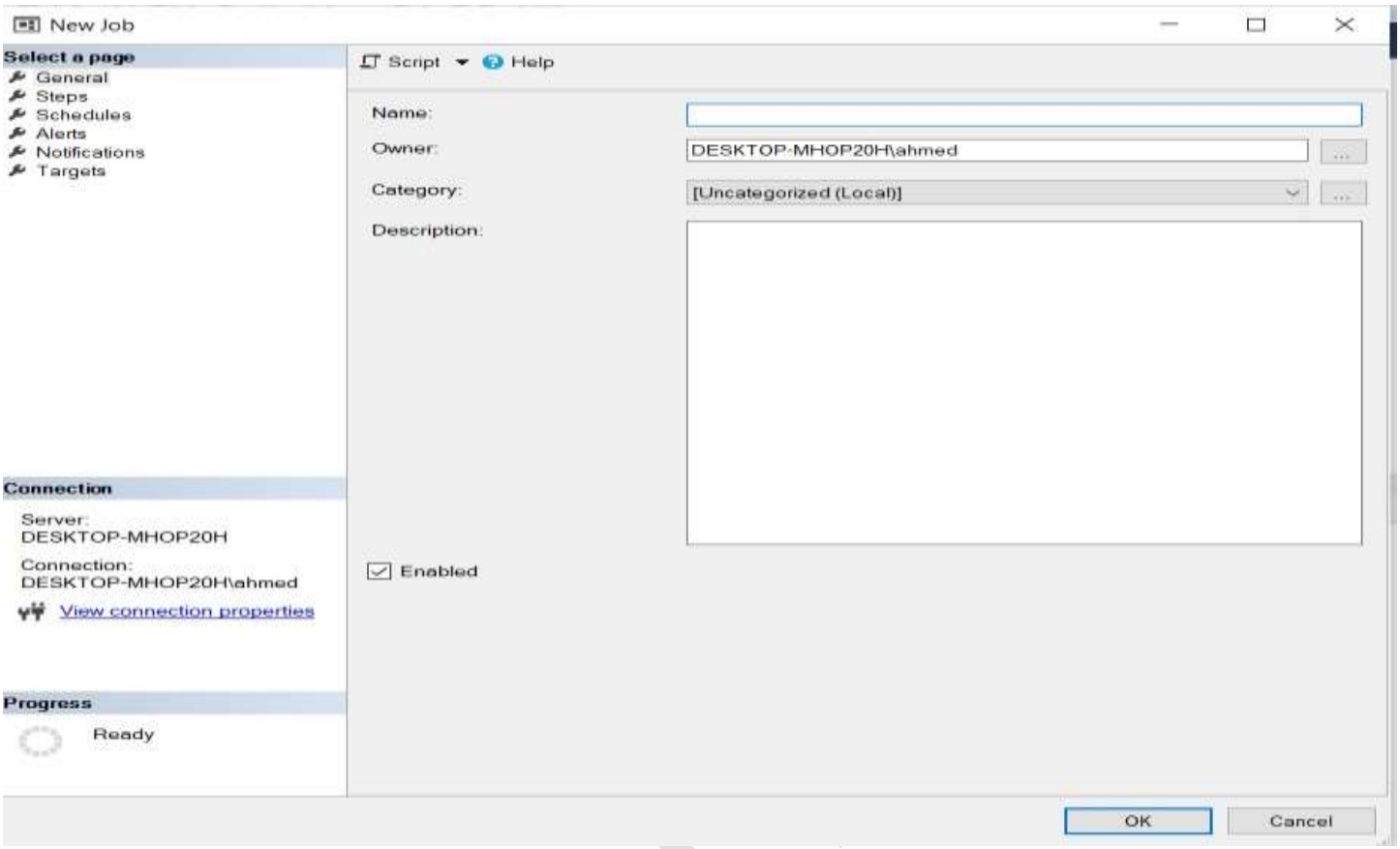
- الـ Job هو Query بيتنفذ مع وجود وقت معين

- إزاى عمل الـ Job :

Server (main Service) → SQL Server Agent (Child service) → Job



- هتظهر الـ **Job wizard** :



- من الـ Steps هقدر أكتب الـ backup Command اللى هعمله

- من الـ Schedule هقدر أحدد الـ Job Scheduling

*** Identity :**

- المعروف إن خاصية الـ Identity على الـ column

(لا بقدر أكتب فيها Data لأنه هو automatic بيـ Fill الـ Identity values)

- لو عملت Table مكون من 3 Column
(الأول Id وعليه Identity – التانى Name – التالت Address)

وعملت Insert لـ 10 values + عملت Delete لـ Records من Id = 3 حتى Id = 8

(اللى هيتبقى فى الـ Table هما Id (1,2,9,10))

عملت Insert لـ record جديد هيبقى الـ Id = 11

-كدا فيه Identity Gap (قيم الـ Identity مش مترتبة)

- الحل : أعدل فى الـ Identity (عن طريق إن أستخدم الـ Syntax الأتى :

```
set Identity_Insert Table_Name On \ Off
```

الـ Default بتاعها Off)

- لو عملت Insert هيبدأ من الرقم موجود فى الـ Identity

- لو عاوز أعمل reset لـ Identity هستخدم الـ syntax :

```
DBCC Checkident ( Table_Name , RESEED , 1)
```

*** Types of Insert :**

1) Simple Insert (one Insertion)

2) Insert Construct (multiple Insertion with the same query)

3) Insert based on select

4) insert based on Execute (SP)

5) Bulk Insert

الـ Bulk Insert :

- هو إن بجيب data من File على الـ hard disk وأرميها فى الـ Table

- شرط الـ File يكون Delimited file

- الـ Syntax :

```
BULK INSERT Table_Name
FROM 'path of File '
WITH (
    FIELDTERMINATOR = ',',   -- Specify the field terminator
    ROWTERMINATOR = '\n',    -- Specify the row terminator
    FIRSTROW = 2             -- Specify the first row to start
                             importing (optional)
);
```

*** SnapShot :**

- تشبه الـ Backup ولكنها أخف بكتير من الـ Backup

- بيتقال عليها Read only Database

- كأنها معاها pointers لكل Database Row فى الوقت اللى عملت فيه الـ snapshot

- الطبيعى إن مفيهاش data ولكن ممكن يبقى فيها data فى حالة

(إن عملت Insert \ Update \ delete على record من الـ table هياخد الـ Record كـ Cut ويحطها فى الـ Snap shot as a data not pointer)

- ممكن أعمل جملة الـ select * from table و أعملها run على الـ DB وعلى الـ Snapshot (هيطلعوا نفس الناتج)

- الـ syntax :

```
CREATE DATABASE database_snapshot_name
ON
(
    NAME = MDf_file_name,
    FILENAME = 'os_file_Name(path)'    -- file extension (.ss)
)
AS SNAPSHOT OF source_database_name;
```

- هيـ Run على الـ Master DB

- ينفع أستخدم الـ Snapshot كـ restore ولكن بشروط :

1) يكون نفس الـ server

2) الـ Database سليمة مش Corrupted

```
RESTORE DATABASE MyDatabase FROM DATABASE_SNAPSHOT = 'MySnapshot';
```

*** SQLCLR (SQL Command Language Runtime) :**

- طريقة للربط الـ SQL بـ C#

- تقدر إنك تعمل حاجات موجودة فى الـ SQL بإستخدام الـ C# وتستخدمها فى الـ SQL Database

- فايدتها :

- الـ C# غنية بالـ Libraries اللى بتساعد كتير

- الـ Runtime Engine الخاص بـ C# أسرع من الـ SQL

- مثلا عاوز أعمل Function أو New datatype (هستخدم الـ SQLCLR)

- لازم أفعل الـ SQLCLR فى الـ SSMS :

```
sp_configure 'clr enabled', 1;
RECONFIGURE;
```