Day 7

*** User defined Function:**

- زى ما فيه built in function بستخدمها فى الـ SQL ، تقدر تـ Create your own function

- الـ main Advantage من إستخدام الـ Function هيا الـ flexibility بحيث أقدر أستخدمها أكتر من مرة فى كذا مكان (بدل ما أكتب الـ Query script فى كل مرة أحتاج فيها الـ Logic بتاع الـ Function)

- عشان أقدر أ Create my Own Function أنا محتاج أعرف الـ

try & catch \ transaction \ script \ while loop \ if statement \ variables

---

*** Variables :**

- الـ Variable ماهو إلا Temporary memory Storage (مكان مؤقت فى الـ memory)

- الـ Variable Types :

1) Local :

- بيبقى على مستوى :

- الـ batch (مجموعة من الـ Highlight queries اللى هعملها run)

- الـ Function

- الـ Stored Procedure

- بيبدأ بـ @

2) Global :

- بيبقى عبارة عن مجموعة من الـ Variables موجودة فى الـ SQL Server بقدر أستخدمها للحصول على Values لكن مقدر أ update أو أغير قيمتها

- بيدأ بـ @@

- أقدر أ assign قيمة الـ Global Var فى الـ Local ولكن العكس غير صحيح

Local_Var = Global_Var → Correct (✓)

Global_Var = Local_Var → Incorrect (X)

Local Variable Declaration :

- الـ Syntax المستخدم فى الـ Declaration:

```
Declare @Var_Name DataType
```

- الـ initial Value بـ NULL

- عشان أقدر أ Assign Value هستخدم واحدة من الـ 4 طرق :

1) أستخدم الـ Set و أ Assign Static Value

```
Set @X = 10 (Static Value)
```

2) أستخدم Select و أ Assign Static Value

```
Select @X = 100 (Static Value)
```

3) أستخدم Column Value من Table عن طريق الـ Select

```
Select @X= Column_Name
from Table_Name
Where Condition
```

هياخد قيمة الـ Column_name اللى محددها بالـ where Clause ويـ assign الـ Value فى X

4) أستخدم Column value من Table عن طريق الـ Update

(ودا بيكون إن أنا بعمل Update لـ Record معين أنا بحدده فبالتالى هو شايف الـ Record ككل وممكن وأنا بـ Set Column أستخدم قيمة أى Column فى الـ Variable)

```
Update Table_Name
Set Column_1 = Value , @X = anyColumValue
Where Condition
```

- عشان أعرض الـ Local Variable بستخدم الـ Syntax

```
Select @X
```

- لازم أخلى كل الـ Queries اللى بتستخدم الـ Local Variable فى نفس الـ Batch

ودا عشان الـ Variableهو Memory Storage فهو موجود فى الـ Run Time بس على عكس الـ values اللى فى الـ Tables (ممكن أخزن قيمة الـ Variable فى Table لو محتاجها على طول)

*** Global Variable :**

- بيبدأ بـ @@
- مقدرش أعمله Declare لأنه already موجود فى الـ SQL Server
- مقدرش أعمله assign
- بقدر أستخدم الـ values اللى متخزنة فيه بس (كـ Display أو أ Assign local Variable)
- بعض الـ Global Variables :

```
Select @@servername
```

- بتـ Return Server name

```
Select @@rowcount
```

- بتـ Return عدد الـ rows اللى affected بأخر Query اتعمله Run

---

```
Select @@version
```

- بتـ Return الـ SQL Server Version (مفيدة إنك تعرفها لأن بعض الـ Queries مش بتشتغل على كل الـ SQL Server Versions)

```
Select @@error
```

- بتـ Return الـ Error Number لأخر Run Query
- لو مكنش فيه Error بيـ Return 0
- لو كان فيه Error بيـ Return Error Number

Select @@identity

- بتـ Return أخر Identity أتعمله Creation فى الـ Table من أخر Insert حصلت

- لو مفيش Identity فى الـ Table بيـ Return NULL

Some of Tips in local Var :

- فى حالة لو أنا إستخدمت Select Query without where clause
(المفروض هيرجع Array of Values والـ variable هيخزن قيمة واحدة بس وهى أخر قيمة)

- فى حالة لو أستخدمت wrong where clause (يعنى مثلا where id = 100 ومفيش record فيه الـ ID = 100) فى الحالة دى الـ batch هيـ Run عادى ولكن قيمة الـ Variable هيحتفظ بأخر قيمة هو خدها فى الـ Batch (سواء كان Initial value او NULL او Assigned Value)

- مينفعش أستخدم الـ Select @X = Value , Column (بمعنى إن أعرض واعمل assign فى نفس الـ Select)

لو عاوز أستخدم Variable يخزن Array of values ساعتها هـ Declare الـ Variable من النوع Table Datatype

Declare @Var_Name table (Column_Name DataType , …… etc)

وهستخدم الـ Insert Based on select عشان الـ assign Valuesفى الـ Table

---

- ممكن أستخدم الـ Variable مع الـ Top (Var) وساعتها هتبقى Dynamic Top

---

- ممكن أستخدم الـ Query as a string كـ Input فى الـ Execute Function

وهيتم عمل الـ Execution عادى لو مفيش Error

Execute (“select * from Table_Name”)

- عبارة عن Dynamic Query

*** Flow Control Statements :**

- If

- Begin

- End

- If exists \ If not exists

- while

- continue

- break

- case

- iif

- wait for

- choose

*** if Statement :**

ـ ممكن أستخدمها فى حالة أنا عندى فى شرط معين أنفذ Query\ies وفى حالة عكس الشرط أنفذ Query\ies

ـ ممكن يبقى عندى كذا Condition وفى الحالة دى هستخدم (If \ Number of Else If \ Else)

Syntax If \ Else:

```
If condition
    Begin
    Queries
    End
Else
    Begin
    Queries
    End
```

Syntax of If \ Else if :

If condition

Begin

Queries

End

Else if condition (ممكن تتكرر حسب عدد الشروط)

Begin

Queries

End

Else

Begin

Queries

End

- الـ Begin والـ End هما بدل لـ { } فى الـ Programming

- ممكن متكتبش Begin و End فى حالة هتكتب One Query بس

*** IF Exists :**

- فى حالة لو أنا عاوز أعمل Check على حاجة موجودة عندى أو لا (مثلا Table_name مثلا بدل ما أستخدم الإسم تانى فى Creation ويدى Error فأنا بتفادى الـ Error وبسأل الأول عن وجوده أو لا)

- Syntax :

If exists (Checking Query 'select something ')

Begin

Queries

End

Else

Begin

Queries

End

- من الأخر هستخدم Query انه بـ retrieve الحاجة اللى محتاجها :

- لو رجع قيمة فعلا والحاجة موجودة فالـ Query هيبقى True

- لو مرجعش قيمة بالتالى هينفذ الـ Else

If not Esists :

- نفس فكرة الـ If Exist ولكن بيتعامل مع العكس

- لو الـ Checking Query :

مش بيـ retrieve حاجة من الـ DB عندى فساعتها هيـ return True

لكن لو موجود وبيـ retrieve حاجة من الـ DB بالتالى هيفذ الـ Else Queries

- ممكن أستخدمة فى فكرة الـ Parentوالـ Child (أتأكد إذا كان للـ parent دا Childes فى الـ Relation-Tables)

- Syntax :

If not exists (Checking Query 'select something ')

Begin

Queries

End

Else

Begin

Queries

End

*** Try \ Catch :**

- بتستخدم فى حالة إنك عندى Query\ies ممكن يبقى فيهم Error بس إنت مش عارف تحدد الـ Error جاى منين

- الأفضل إنك متعملش run لـ Queries فيها Errors عشان الـ APP ميضربش منك

(تستخدم طرق الـ Checking مع الـ queries عشان تضمن)

- Syntax :

Begin try

Queries

End Try

Begin Catch

Queries

End Catch

- ممكن أستخدم مجموعة من الـ built in Functions اللى بتستخدم لتعريفك الـ Error مع الـ Select

1) ERROR_LINE() : بتـ return رقم الـ line اللى فيه الـ Error

2) ERROR_NUMBER() : بتـ return رقم الـ Error فى الـ SQL

3) ERROR_MESSAGE() : بتـ return الـ Error Message

*** While Loop :**

- بتستخدم فى حالة إنك عاوز تكرر جزء معين من الـ SQL Queries على Iterations

- بستخدم Local Variable مع الـ while عشان أقدر أعمل iterations

- Syntax :

```
Declare @iterator int
While @iterator < Value
    Begin
    Queries ( include Increment Iterator)
    End
```

- مع العلم :

Continue : بتروح لـ Next iteration condition

Break : بتنهى الـ Loop وتخرج منها

*** Case :**

- نفس فكرة الـ If else بس Within query

Syntax :

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    ...
    ELSE defaultResult
END
```

*** iif :**

- هى عبارة عن Built in function بتستخدم فى الـ Flow Control

- زى الـ Ternary Operator فى الـ programming

Syntax :

```
IIF (condition, true_value, false_value)
```

*** wait For :**

- بتستخدم لعمل Delay بوقت معين لـ Runned Script

-

Syntax :

```
WAITFOR DELAY 'time_to_wait';
```

*** Batch Vs Transaction vs Script :**

- الـ batch :

هو عبارة عن مجموعة من الـ highlight queries اللى بحددهم عشان الـ Run

(ملهومش علاقة ببعض ولا بيعتمدوا على بعض)

- الـ Script :

هو عبارة عن queries مينفعش تـ Run مع بعض فى نفس الـ Batch

(زى الـ create Rule و الـ sp_bindrule)

- بفصل بينهم بـ go (فى حالة عاوز أعملهم run فى نفس الـ batch)

الـ transaction :

هو عبارة عن مجموعة من الـ Queries إما تتنفذ كلها مع بعض أو ميتنفذش جاجة
(زى مثلا عمليات التحويل البنكى)

- Single unit of business work

*** Query file Vs Ldf file Vs Mdf file :**

- الـ Query file : المكان اللى بكتب فيه الـ SQL Queries

- الـ Ldf file : دا الـ Log file الخاص بـ DB Transaction اللى عملها بالـ Queries

- الـ Mdf file : دا الـ Metadata\ Data file بيبقى فيه كل محتويات الـ DB obj والـ data اللى فيها

- لما بعمل Insert Query مثلا فى الـ Query file وأعمل execute أو F5 بالتالى الـ query بيروح الـ Log file يتخزن As an implicit Transaction

- فى الـ log file بيبقى بيبدأ الـ transaction بعدين يكتب إنه عاوز ينفذ جملة Insert بعدين يروح ينفذها على الـ Mdf File

- فى الـ mdf file لو اتنفذت واتعمل التعديل المطلوب ، الـ server بيرجع يكتب فى الـ log file

- بيكتب Commit وينتهى الـ transaction

- لو حصل Error الـ server بيكتب Rollback

*** Explicit Transaction :**

- دى فى حالة إن يبقى فيه مجموعة من الـ queries أنت عاوزهم فى نفس الـ Transaction جوا الـ Log file (قبل كدا فى الـ implicit كان كل query ليه الـ Transaction فى الـ log الخاصة بيه)

- دلوقتى بقا مجموعة من الـ queries موجودين فى نفس الـ transaction فى الـ Log

- لو عملت Explicit Transaction لـ 3 Queries مثلا (Insert\Update\Delete) فيه كذا Scenario فى الحالة دى :

1) فى خلال التنفيذ مثلا حصل Server Crash وكان هو نفذ الـ insert والـ update مثلا

فى الحالة دى هيتم عمل automatic Rollback لأنه منفذش الـ 3Queries مع بعض

2) فى حالة إن الـ insert والـ update تمام ولكن حصل error فى الـ delete

فى الحالة دى هيبقى متوقف ع أنك عامل Commit ولا Roll back فى الـ Query

لو عامل commit هيعمل commit لـ queries اللى اتنفذت بس

لو عامل Rollback هيتم عمل Rollback لكل اللى اتنفذ (حتى لو كله إتنفذ بدون eror)

- عشان أحل المشكلة دى بستخدم Try و Catch دايماً مع الـ Explicit Transaction عشان أقدر أحقق الـ Logic المطلوب

- Syntax :

```
Begin Try
    Begin Transaction
    Queries
    Commit
End Try
Begin Catch
    Roll back
End Catch
```

- فى الـ syntax دا إنت محدد إن لو فيه error فى أى query على طول روح لـ Roll back ولو مفيش error فمعنى كدا إن الـ Queries كلها إتنفذت بدون error فبالتالى إعمل commit (ودا يحقق الـ Logic بتاع الـ transaction)

---

* Functions :

- فيه نوعين من الـ Functions:

1) Built in function

2) User Defined Function

*** Built in Functions :**

- الـ NULL Functions:

Isnull (Column name , replacement)

ودى function بتـ check إذا كان الـ column بـ null بيـ replace الـ Null بـ replacement

(ممكن يكون Value من نفس الـ DataType أو Other Column)

Coalesce (Column_Name , Replacement 1 , Replacement 2 ,… Replacement N)

نفس فكرة الـ Is null ولكن multiple Replacements

Nullif (Column)

بترجع إذا كان الـ column المستخدم دا بـ Null ولا لا (True or false)

- الـ System Functions :

db_name()

بتـ return الـ Database Name اللى انت شغال عليها

suser_name()

بتـ return الـ User Name اللى عامل login وبيستخدم الـ queries

---

- الـ Converting DataType Functions :

Convert (DataType , Expression(Value or Column) , [Style])

- دى Function بتحول الـ DataType وممكن كمان أستخدم الـ Parameter التالت فى تحديد الـ Style (فى الـ date فيه أرقام بتدل على شكل الـ Format)

Cast (Expression as DataType)

- نفس فكرة الـ Convert ولكن الفرق إن cast مش بحدد الـ Style زى الـ convert

Format (getdate() , 'yyyy – mm – dd ')

- بتستخدم غالباً مع الـ date & time Formating كطريقة أسهل من إن أعرف رقم الـ style وأستخدم الـ convert Function

---

- الـ String Functions :

- Substring (input_string , start_point , length)

بتستخدم عشان تـ extract جزء من الـ string معين

- Upper (input_string)

بيخلى الـ String characters عبارة عن Upper case

- Lower (input_string)

بيخلى الـ String characters عبارة عن Lower case

- Length (input_string)

بيـ Return الـ Number of Characters اللى موجودين فى الـ Input String

---

الـ Date Function :

- Getdate ( )

بتـ return الـ Current system date & time

- Year (date_ expression)

بتـ Extract الـ Year part من الـ date

- Month (date_ expression)

بتـ Extract الـ month part من الـ date

- Day (date_ expression)

بتـ Extract الـ day part من الـ date

الـ Aggregate Functions :

- Count (column_ name)

بتـ return الـ number of records فى الـ column (Null Exception)

- Sum (column_ name)

بتـ return الـ Sum لـ records فى الـ column (Null Exception)

- Avg (column_ name)

بتـ return الـ Average لـ records فى الـ column (Null Exception)

- min (column_ name)

بتـ return الـ Minimum Value لـ records فى الـ column (Null Exception)

- Max (column_ name)

بتـ return الـ Maximum Value لـ records فى الـ column (Null Exception)

الـ maths Functions : هيا عبارة عن Functions بتتطبق على الـ Numeric data

1) الـ operators الموجودة فى الـ SQL

- 'select Numeric_column + Numeric_Column as Sum_result ' → Addition

- 'select Numeric_column – Numeric_Column as difference ' → Substraction

- 'select Numeric_column * Numeric_Column as product ' → Multiplication

- 'select Numeric_column / Numeric_Column as quotient' → Division

2) الـ Exponential Functions :

- Power (base , exponent)

Or

- Base ^ exponent

3) الـ square root function :

- Sqrt (number)

بتـ return الجذر التربيعى لـ Number

4) الـ Trigonometric Function :

- SIN (Angel)

بتـ return الـ Sine Value لـ Angle

- COS (Angel)

بتـ return الـ Cosine Value لـ Angle

- TAN (Angel)

بتـ return الـ Tanget Value لـ Angle

- الـ Ranking Function :

Row_Number () over (order by Column)

بترتب الـ Table بالـ column وكل Record معاه Rank number Ascending from 1

Dense_Rank () over (order by Column)

بترتب الـ Table بالـ column وكل Record معاه Rank Asc from 1 (الـ values المتشابهه معاها نفس الـ Rank)

Rank () over (order by Column)

بترتب الـ Table بالـ column وكل Record معاه Rank Asc from 1 (الـ values المتشابهه معاها نفس الـ Rank + الـ Next rank عبارة عن الـ Previous rank ومتضاف عليه عدد الـ Columns اللى فى الـ previous rank "يعنى R2 فيه 3 records يبقى Next R عبارة عن 5"

NTiles (Number) over (order by Column)

بترتب الـ table وتقسمه لـ "Groups" Number of Tiles بحيث النقص يكون فى أخر Groups والنقص ميزدش عن 1

الـ Logical Functions :

Iif (condition , True_value , False_value)

الـ Immediate if شبيهه لـ ternary operator فى الـ programming

(بتـ return الـ True value فى حالة الـ condition is true و

بتـ return الـ false Value فى حالة الـ Condition is false)

Choose (index , Val_1 , Val_2 , ……. ,Val_N)

الـ choose يا عبارة عن Function بتكتب الـ index بيـ return الـ Corresponding value

(يعنى لو الـ index = 2 هيـ return الـ Val_2 .... وهكذا "ممكن الـ index يكون Variable")

- الـ Windowing Function

هى نفس فكرة الـ Ranking Function بس الـ ranking كانت بتظهرلك أرقام وبعد كدا إنت بتستخدم الأرقام دى فى Subqueries إنك تختار الـ rank اللى أنت عاوزه

لكن الـ windowing بتعمل الكلام دا Implicitly و بتـ return 4 Function values

(لو فكرت تـ Implement الـ Function Logic هتستخدم Subqueries كتير)

LEAD (Column) over (order by Ordering _Column)

-بترتب الـ Table بـ Ordering_Column و تعرض الـ Column value لـ Next record الخاص بالـ Current Record

LAG (Column) over (order by Column)

-بترتب الـ Table بـ Ordering_Column و تعرض الـ Column value لـ Previous record الخاص بالـ Current Record

First_Value (Column) over (order by Column)

-بترتب الـ Table بـ Ordering_Column و تعرض الـ Column value لـ first Record فى الـ Table

Last_Value (Column) over (order by Column)

-بترتب الـ Table بـ Ordering_Column و تعرض الـ Column value لـ Last Record فى الـ Table

---

- كل الـ built in function بيطلق عليها Scalar Functions (عشان بتـ Return single value)

*** الـ User Defined Functions :**

- بيتم تخزينهم فى الـ Programmability Folder

- عبارة عن 3 Types وكل type ليه الـ syntax الخاص بيه

- بيتقسموا لـ 3 Types بناء على الـ return value

- نوع الـ function يعتمد على الـ return value

- الـ Function body بيبقى فيه Select statements فقط (من الأخر أى queries غير اللى تعدل فى Tables)

- ممكن أعمل الـ function ليها parameters أو لا (حسب منا عاوز)

- الـ 3 Types of user defined functions :

1) Scalar Function (بتـ return single value)

2) Inline Table-Valued Function (بتـ return Table)

3) Multi-Statement Table-Valued Function (بتـ return Table)

---

1) Scalar Function :

- بتـ return Single Value

*** Syntax of Scalar Function (signature + Body) :**

```
Create Function Function_name (@Local_Variables)
Returns DataType
Begin
    Queries
    return Value \ Variable_holding_value
End
```

*** Invoking of Scalar Function :**

- لازم تحط إسم الـ Schema قبل إسم الـ function حتى لو dbo

(لأنك لو معملتش كدا هيعتبرها Built in Function مش UDF)

```
Select Schema_name.Function_name (@local_variables_Values)
```

---

2) Inline Table-Valued Function :

- بتـ return Table

*** Syntax of Inline Table-Valued Function (signature + Body) :**

```
Create Function Function_name (@Local_Variables)
Returns table
As
return
(
Queries ( select statements only )
)
```

*** Invoking of Inline Table-Valued Function :**

Select * \ Columns From Schema_name.Function_name ()

---

3) Multi-Statement Table-Valued Function :

- بتـ return Table

**- الفرق بينها وبين الـ Inline هو الـ Function body **

(فى الـ Inline بتستخدم Select بس

فى الـ Multi-Statement بتستخدم (Select + Logic (if ….etc)

*** Syntax of Multi-Statement Table-Valued Function (signature + Body) :**

```
Create Function Function_name (@Local_Variables)
Returns  @table_name table
            (
              [Column_Name]  DataType
            )
As
Begin
      Queries ( mostly used “Insert based on select ”  + Logic )
      return
End
```

*** Invoking of Multi-Statement Table-Valued Function :**

Select * \ Columns From Schema_name.Function_name ()

- ممكن أستخدم أى Logic (if else \ if else if …… etc)

- الـ insert based on select هيا insert statement بس مش بتعدل فى Table هيا بتعمل copy لـ Result Set اللى جت من الـ Select فى الـ Table اللى فى الـ Insert