Day 9 part 2

*** Write a report to display difference between the following objects in SQL Server :**

1- Trigger Vs Stored procedure :

Comparison	Triggers	Stored Procedures
Purpose	- بيستخدم فى حالة إن عاوز أعمل SQl Queries بيتم تنفيذهم فى حالة حدوث Event معين على Table or View - Event-Driven	- بيستخدم فى حالة الـ reusable logic لتنفيذ Repetitive Task (عندى SQL Queries بيتكرر إستخدامها) - Repetitive Tasks - Pre-compiled (Performance wise)
Execution	- بيتم تنفيذ الـ Triggers بشكل Automatic فى حالة حدوث Events على الـ Table أو الـ View - الـ Events هى : 1) Insert 2) Update 3) Delete - Automatic Implicit invoking based on specific events occur on Table or View	- بيتم تنفيذ الـ Stored Procedurs فى حالة إن أعمل لها أنا Invoking - Explicit Invoking (run when Invoked)
Scope	Specified with table or view	- Independent Specification
Types	1- Before Trigger : - بيتم تنفيذه قبل حدوث الـ Event على الـ Table\View 2- After Trigger : - بيتم تنفيذه بعد حدوث الـ Event على الـ Table\View 3- Instead of Trigger : - بيمنع تنفيذ الـ Event على الـ Table \ view	1- Built in SP : - عبارة عن Built in procedures موجودة فى الـ SQL 2- User-Defined SP : - عبارة عن procedures الـ user بيعرفها بمعرفته 3- Triggers : - عبارة عن Special type of SP بتـ Run Automatically فى حالة حدوث Event على الـ Table \ View
Parameter Handling	- Don't accept Parameters	- Can accept 3 types of parameters : 1) input 2) output 3) input\output
Return Values	- Don't return Values	- Can return Values or Result sets
Use Cases	1- Enforcing Referential Integrity : - ممكن أستخدمها فى كـ constraints على الـ Insert \ Update \ Delete 2- Auditing : - تسجيل معلومات عن الـ Events اللى حصلت على الـ Table\View (user name \ Time ….etc) وكمان الـ table قبل وبعد الـ Event عن طريق الـ : Inserted Table – Deleted Table	1- Encapsulating Complex logic : - كتابة Complex Logic داخل الـ SP 2- Repetitive Tasks : - Logic بيتكرر فممكن أنفذه بإستخدام الـ SP ودا بيخلى الـ Code يبقى readable أكتر 3- Performance : - بيسرع الـ Performance لأنه Pre-Compiled

Syntax	`Create Trigger Trigger_Name`
	`On (Table_Name\View_Name)`
	`(Before\After\Instead of)`
	`(Insert\Update\Delete)`
	`as`
	`Queries`
		`Create Procedure SP_Name`
		`@Paramenters ParameterDataType`
		`as`
		`Queries`

2- Function Vs Stored Procedures :

Comparison	Function	Stored Procedures
Purpose	- بيستخدم فى حالة الـ reusable logic لتنفيذ Repetitive Task (بيكون Select Queries بس – أو لو هستخدم Calculations)	- بيستخدم فى حالة الـ reusable logic لتنفيذ Repetitive Task (عندى SQL Queries بيتكرر إستخدامها) - Repetitive Tasks - Pre-compiled (Performance wise)
Body	- Only Select Statements (Queries Except DML Queries)	- Any Type of Queries
Parameter Handling	- Can accept Input parameters :	- Can accept 3 types of parameters : 1) input 2) output 3) input\output
Return Value	- Can return Values or Result sets	- Can return Values or Result sets
Invoking	- Explicitly Invoked	- Explicitly Invoked
Types	- الـ Function types بتتحدد بنوع الـ returned Value 1- Scaler Function : - بتـ return Single Value 2- Inline Table-Valued Function : - بتـ return Table - الـ function Body عبارة عن Select Statements Only 3- Multi-Statement Table-Valued Function : - بتـ return Table - - الـ function Body عبارة عن Select + Logic (if …etc)	1- Built in SP : - عبارة عن Built in procedures موجودة فى الـ SQL 2- User-Defined SP : - عبارة عن procedures الـ user بيعرفها بمعرفته 3- Triggers : - عبارة عن Special type of SP بتـ Run Automatically فى حالة حدوث Event على الـ Table \ View

- الـ SP والـ Function بيستخدموا فى نفس الأغراض تقريباً ولكن الفرق إن :

1) الـ SP بتكون More Performance من الـ Functions

2) الـ SP بتـ Contain any type of Queries لكن الـ Function بتـ Mostly Contain Select Queries

3- Drop Vs Delete :

Comparison	Drop	Delete
Query Type	**- Data Definition Language (DDL) Query**	**- Data Manipulation Language (DML) Query**
Purpose	**- Remove Entire Database Object (Data and Metadata)**	**- Remove Specific or Entire Data from Database object (Data only)**
Scope	**- operates at Schema Level**	**- Operates at Row Level**

Syntax	`Drop Table Table_Name`	`Delete From Table_Name`
		`( -- specific Deletion based on where clause but Table is still remains in Database)`
		`Where Condition`
		
		`Delete From Table_Name`
		`(--Entire Deletion but Table is still remains in Database)`

4- Select Vs Select into :

Comparison	Select	Select into
Query Type	**- Data Query Language (DQL) Query**	**- Data Definition Language (DDL) Query**
Purpose	**- Display data from One or more Table As Result Set**	**- Create Table (Metadata + Data) based on Selection from Table \ Result set**
Use cases	**- Data Querying** **- Data Analysis** **- Data Reporting**	**- Backup Tables** **- Temporary Tables** **- Data Migration**

Syntax	`SELECT Column1, Column2, ...`
	`FROM Table_Name`
		`SELECT column1, column2, ...`
		`INTO New_Table_Name`
		`FROM Table_Name ...`

5 – DDL Vs DML vs DCL vs DQL

Comparison	DDL	DML	DCL	DQL
Abbreviation	Data Definition Language	Data Manipulation Language	Data Control Language	Data Query Language
Purpose	Define | Manage Database Structure (MetaData)	Manage and Manipulate Data within Tables	Control Access and Permissions on Database Objects	Retrieve Data From DataBase
Common Commands	- Create - Alter - Drop - Truncate - Rename	- Insert - Update - Delete - Merge	- Grant - Revoke	- Select
Use Cases	- Creating - Altering - Deleting Tables and other Database Objects	- Inserting - Updating - Deleting Data From Tables	- Granting - Revoking User Permissions	- Querying Data

6 – Inline Table-Valued Function Vs Multi-Statement Table-Valued Function :

Comparison	Inline Table-Valued Function	Multi-Statement Table-Valued Function
Definition	Single Select Statement	Block of Multiple Select Statements
Performance	More Efficient Due to Simple Execution	Can be less Efficient due to handling Multiple Statements and Temporary Tables
Use Cases	Simple logic Expressed in a single Select Statement	More Complex Logic Requiring Multiple Steps
Syntax	`CREATE FUNCTION Function_Name (@parameters Datatype) RETURNS TABLE AS RETURN (SELECT ...)`	`CREATE FUNCTION Function_Name (@parameters Datatype) RETURNS @Table_Name TABLE ( columns… ) AS BEGIN INSERT INTO @Table_Name SELECT … RETURN END`

7- Varchar(50) Vs Varchar(Max)

Comparison	Varchar(50)	Varchar(Max)
Maximum Length	- 50 Characters (ability store from 0 to 50 Char)	- Up to 2,147,483,647 Characters (ability to store from 0 to 2,147,483,647 Characters)
Use Cases	Predictable Small-sized Text	Unpredictable Large-sized Text
Storage	N + 2 Bytes Where N → Actual Length	- N+2 Bytes for Small Data - Larger Values Stored out of row Up to (2GB)
Flexibility	Less Flexibility Limited to 50 Character	High Flexibility
Performance	Faster for fixed small sizes	Slower for very large Sizes

8- SQL And Windows Authentication

Comparison	SQL Authentication	Windows Authentication
Authentication Method	- Created By Database Admin (Windows Admin or SQL Server Admin) - Use Specified User Name And Password to get login in Database server	- Actually Windows User is the Admin for Database Server
Access Scope	- Admin Customize the Scope of Access (Database , Tables , Schema) to every SQL User	- Entire Server

9- Inline Table-Valued Function Vs View

Comparison	Inline Table-Valued Function	View
Definition	User Defined Function returns a table data type	- virtual Table based on the result set from Select Query
Use Cases	Repetitive Logic Tasks	- Simplify Complex Select Queries - Provide Abstraction (Metadata are unknown)
Security for Metadata	- No Security	- Provide Abstraction
Execution	- Executed every time	- Precompiled and Stored in the database

Syntax	`CREATE FUNCTION Function_Name (@parameters Datatype)`
	`RETURNS TABLE @Table_Name`
	`AS`
	`RETURN`
	`SELECT ...`
		`CREATE VIEW View_Name`
		`AS`
		`SELECT ...`

10 – Identity Vs Unique Constraints

Comparison	Identity	Unique
Purpose	**Automatically Generate Unique Identifiers**	**Ensure Uniqueness of Column's Values Regardless of being Primary Key**
Use Cases	**- Primary Key** **- Unique Identifier**	**Columns require to have unique values Regardless of being Primary Key (Business wise)**
Characteristics	**- Automatically Managed by Database System**	**- Explicitly Defined Constraint**
Syntax	`PK_Column INT IDENTITY(1,1) PRIMARY KEY`	`Non_PK_Column VARCHAR(100) UNIQUE`