# Mohab Ahmed Mohamed Wafaie

1. trigger and stored procedure:

**Trigger**

* Executes automatically when an INSERT, UPDATE, or DELETE occurs.
* Used for enforcing rules, logging, or auditing.
* Cannot be called manually.
* Cannot return values.
* Types include AFTER, INSTEAD OF, and DDL triggers.

**Stored Procedure**

* Executes manually using the EXEC command.
* Used for data processing, business logic, or reporting.
* Can accept input parameters and return output.
* Provides full control over execution

1. stored procedure and functions

**Stored Procedure**

* Executes a set of SQL statements manually using the EXEC command.
* Can perform DML operations like INSERT, UPDATE, DELETE.
* Can return multiple result sets and output parameters.
* Allows exception handling using TRY...CATCH.
* Does not need to return a value, but can return values if needed.

**Function**

* Returns a single value or a table and is called within a SQL expression.
* Cannot perform changes to the database (i.e., no INSERT, UPDATE, DELETE).
* Must return a value (scalar or table).
* Cannot use TRY...CATCH for exception handling.
* Used mainly for calculations or reusable logic inside queries.

1. drop and delete statement

**DELETE**

* Removes rows from a table based on a condition (can use WHERE).
* Does not remove the table structure; the table remains in the database.
* Can be rolled back if used inside a transaction.
* Can be filtered to delete specific rows.
* Logs each deleted row, making it slower for large datasets.

**DROP**

* Completely removes the table (or other object like a view or database) from the database.
* Deletes both data and structure.
* Cannot be rolled back once executed.
* Does not support WHERE clause — it drops the entire object.
* Faster, but more destructive than DELETE.

1. select and select into statement

**SELECT**

* Retrieves data from existing tables.
* Does not create a new table.
* Commonly used to view or analyze data.
* Syntax example:  
  SELECT \* FROM Employees;

**SELECT INTO**

* Retrieves data and creates a new table with that data.
* Copies both structure and data from source table(s).
* Useful for creating backups or temporary tables.
* Syntax example:  
  SELECT \* INTO NewEmployees FROM Employees WHERE Department = 'HR';

1. DDL,DML,DCL and DQL

**DDL (Data Definition Language)**

* Used to define and manage database structure.
* Affects schema (tables, views, indexes, etc.).
* Common commands:
  + CREATE
  + ALTER
  + DROP
  + TRUNCATE

**DML (Data Manipulation Language)**

* Used to manage data inside tables.
* Affects table content, not structure.
* Common commands:
  + INSERT
  + UPDATE
  + DELETE
  + MERGE

**DCL (Data Control Language)**

* Used to control access and permissions in the database.
* Manages security and user rights.
* Common commands:
  + GRANT
  + REVOKE
  + DENY

**DQL (Data Query Language)**

* Used to query and retrieve data from the database.
* Focused on data retrieval only.
* Main command:
  + SELECT

1. Table valued and multi statement function

**Inline Table-Valued Function**

* Returns a table directly from a single SELECT statement.
* No BEGIN...END block used.
* Performance is generally better, as it behaves like a parameterized view.
* Cannot contain additional logic or multiple queries.
* Syntax is more concise.

**Multi-Statement Table-Valued Function**

* Returns a table variable defined and populated within the function.
* Uses a BEGIN...END block.
* Can include multiple SQL statements, complex logic, and conditional processing.
* Slightly slower than inline functions due to internal processing.
* Offers more flexibility.

1. Varchar(50) and varchar(max)

**VARCHAR(50)**

* Stores variable-length character data up to 50 characters.
* Best for columns with a known and limited size (e.g., names, codes).
* Uses less storage and offers better performance for small text.
* Enforced length limit: inserting more than 50 characters will cause an error (or truncation, depending on settings).

**VARCHAR(MAX)**

* Stores variable-length character data up to 2^31-1 characters (approx. 2 GB).
* Suitable for large text data, such as documents, descriptions, or logs.
* Can have slightly lower performance, especially when handling large data.
* Allows functions like SUBSTRING, but some operations (e.g., indexing) may be restricted or limited.

1. SQL and windows Authentication

**SQL Authentication**

* Uses username and password managed by SQL Server itself.
* Users must provide credentials when connecting.
* Credentials are stored and verified by the SQL Server.
* Useful when clients are outside the Windows domain or mixed environments.
* Requires separate security management.
* Passwords can be encrypted but are maintained by SQL Server.

**Windows Authentication**

* Uses Windows user accounts or Active Directory for login.
* Relies on Windows security and single sign-on (SSO).
* Users connect automatically if logged into their Windows domain.
* Provides stronger security using integrated Windows protocols.
* No need to store passwords in SQL Server.
* Preferred in corporate environments for ease of management.

1. Inline function and view

**Inline Function**

* A table-valued function that returns a table result from a single SELECT statement.
* Can accept parameters, allowing dynamic filtering and reusable logic.
* Acts like a parameterized view.
* Can be used in queries like a table or joined with other tables.
* Supports modular and encapsulated logic.

**View**

* A virtual table representing the result of a stored SELECT query.
* Cannot accept parameters; always returns the same result set based on underlying data.
* Simplifies complex queries by encapsulating them into reusable objects.
* Acts like a saved query that can be used in SELECT, JOIN, etc.

1. Identity and unique Constraint

**IDENTITY**

* Automatically generates incrementing numeric values for a column (usually used as a primary key).
* Values are generated by SQL Server when a new row is inserted.
* Commonly used for auto-incrementing IDs.
* Syntax example: ID INT IDENTITY(1,1) (starts at 1, increments by 1).
* Ensures uniqueness by default but is mainly for generating sequential numbers.

**UNIQUE Constraint**

* Ensures that all values in a column or a set of columns are unique (no duplicates allowed).
* Can be applied to any column type, not just numeric.
* Does not generate values; values must be provided during insert/update.
* Can have multiple unique constraints per table.
* Helps enforce data integrity by preventing duplicate entries.