**Part2**

**Trigger and Stored Procedure**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Trigger** | **Stored Procedure** |
| **Definition** | Executes automatically in response to table/view events (e.g., INSERT, UPDATE, DELETE). | Precompiled SQL statements executed explicitly by a user or application. |
| **Purpose** | Enforces business rules, data integrity, and responds to data changes. | Encapsulates reusable logic for data manipulation and retrieval. |
| **Execution** | Executes automatically upon triggering events. | Executes explicitly using EXEC or EXECUTE. |
| **Scope** | Tied to a specific table or view. | Independent of tables or views. |
| **Performance Impact** | Can slow down transactions due to automatic execution. | Precompiled, optimized for execution, and generally faster. |
| **Use Cases** | Auditing, logging, enforcing referential integrity, cascading actions. | Batch processing, complex queries, dynamic reports, and workflows. |
| **Syntax Example** | CREATE TRIGGER for event-driven actions. | CREATE PROCEDURE for reusable operations. |
| **Limitations** | Harder to debug, maintenance challenges with complex logic. | Requires explicit calls, not suitable for automatic enforcement. |

**Stored Procedure and Functions**

|  |  |  |
| --- | --- | --- |
| Feature | Stored Procedure | Function |
| **Definition** | Precompiled collection of SQL statements executed explicitly. | A reusable SQL object that returns a single value or table. |
| **Purpose** | Performs complex operations and manipulates data. | Calculates and returns a value or result set. |
| **Execution** | Called using EXEC or EXECUTE. | Invoked as part of a SQL expression (e.g., SELECT, WHERE, etc.). |
| **Return Type** | Can return multiple result sets or no value. | Always returns a single scalar value or a table. |
| **Side Effects** | Can perform modifications to database objects (e.g., INSERT, UPDATE). | Cannot modify database objects; only performs read operations. |
| **Use Cases** | Complex business logic, batch processing, dynamic SQL execution. | Calculations, aggregations, and reusable logic in queries. |
| **Syntax Example** | CREATE PROCEDURE for encapsulating operations. | CREATE FUNCTION for creating reusable calculations. |
| **Limitations** | Cannot be used directly in SQL expressions. | Limited to operations that do not alter database state. |

**Drop and Delete Statement**

|  |  |  |
| --- | --- | --- |
| Feature | DROP Statement | DELETE Statement |
| **Definition** | Removes an entire database object (e.g., table, view, procedure). | Deletes specific rows from a table based on a condition. |
| **Impact** | Completely removes the object and its structure from the database. | Removes only the data but keeps the table structure intact. |
| **Rollback** | Cannot be rolled back once executed (unless within a transaction). | Can be rolled back if enclosed in a transaction. |
| **Use Cases** | Used when the object is no longer needed. | Used to remove specific data while retaining the table. |
| **Performance Impact** | Faster as it directly removes the object. | May take time depending on the number of rows to be deleted. |
| **Syntax Example** | DROP TABLE TableName; | DELETE FROM TableName WHERE Condition; |
| **Limitations** | Irreversible outside of transactions. | Requires a condition to avoid deleting all rows accidentally. |

**Select and Select into Statement**

|  |  |  |
| --- | --- | --- |
| Feature | SELECT Statement | SELECT INTO Statement |
| **Definition** | Retrieves data from one or more tables or views. | Retrieves data and creates a new table with the selected data. |
| **Purpose** | Used to query and display data from existing tables. | Used to create a new table and populate it with the result of a query. |
| **Table Creation** | Does not create a new table. | Automatically creates a new table based on the query result. |
| **Execution** | Only displays the query result; no structural changes to the database. | Creates a table and inserts data in one step. |
| **Syntax Example** | SELECT ColumnName FROM TableName WHERE Condition; | SELECT ColumnName INTO NewTableName FROM TableName WHERE Condition; |
| **Use Cases** | Viewing and analyzing existing data. | Quickly backing up or archiving data into a new table. |
| **Limitations** | Requires a separate CREATE TABLE statement for new tables. | Cannot modify the structure of the new table during creation. |

**DDL, DML,DCL and DQL**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Category | DDL (Data Definition Language) | DML (Data Manipulation Language) | DCL (Data Control Language) | DQL (Data Query Language) |
| **Definition** | Defines and modifies database structure (tables, schemas, etc.). | Handles data manipulation within tables (insert, update, delete). | Manages access and permissions for database objects. | Retrieves data from the database using queries. |
| **Key Commands** | CREATE, ALTER, DROP, TRUNCATE | INSERT, UPDATE, DELETE, MERGE | GRANT, REVOKE, DENY | SELECT |
| **Purpose** | Focuses on the structure and schema of the database. | Focuses on manipulating and modifying data in tables. | Focuses on controlling user access and permissions. | Focuses on retrieving data for analysis and reporting. |
| **Effect** | Affects database objects and their definitions. | Affects the data stored in the database. | Affects security and access rights. | Does not modify data; only retrieves it. |
| **Rollback** | Not applicable for most commands. | Rollback is possible for INSERT, UPDATE, and DELETE. | Not applicable. | Not applicable. |
| **Examples** | CREATE TABLE Employees (ID INT, Name VARCHAR(50)); | INSERT INTO Employees (ID, Name) VALUES (1, 'John'); | GRANT SELECT ON Employees TO User1; | SELECT \* FROM Employees WHERE Name = 'John'; |

**Table Valued and Multi Statement Function**

|  |  |  |
| --- | --- | --- |
| Feature | Table-Valued Function (Inline) | Multi-Statement Table-Valued Function |
| **Definition** | Returns a table as the result of a single SELECT statement. | Returns a table constructed by multiple statements. |
| **Execution** | Executes a single SELECT statement. | Allows multiple SQL statements to build the result set. |
| **Return Type** | Directly returns the result of the SELECT query as a table. | Requires a table variable to store and return the result. |
| **Performance** | Generally faster due to the simplicity of execution. | May be slower due to multiple statements and additional processing. |
| **Use Cases** | Simple transformations and queries. | Complex logic requiring multiple steps to build the result set. |
| **Syntax Example** | CREATE FUNCTION FuncName() RETURNS TABLE AS RETURN (SELECT...); | CREATE FUNCTION FuncName() RETURNS @TableVar TABLE (...) AS BEGIN INSERT INTO @TableVar...; RETURN @TableVar; END; |
| **Limitations** | Limited to single SELECT query logic. | Requires explicit population and handling of the table variable. |

**Varchar(50) and Varchar(MAX)**

|  |  |  |
| --- | --- | --- |
| Feature | Table-Valued Function (Inline) | Multi-Statement Table-Valued Function |
| **Definition** | Returns a table based on a single SELECT query. | Returns a table using multiple statements and complex logic. |
| **Complexity** | Limited to simple queries. | Supports complex logic and operations. |
| **Performance** | Faster due to inline execution plan. | May have slower performance due to multiple statements. |
| **Syntax Example** | CREATE FUNCTION FuncName() RETURNS TABLE AS RETURN (SELECT ...); | CREATE FUNCTION FuncName() RETURNS @Table TABLE (...) AS BEGIN ... END; |
| **Use Cases** | Simple calculations, filtering, or projections. | Complex data transformations and aggregations. |
| **Return Structure** | Directly derived from the SELECT query. | Defined explicitly with DECLARE and manipulated within. |

**SQL and Server Authentication**

|  |  |  |
| --- | --- | --- |
| Feature | SQL Authentication | Windows Authentication |
| **Definition** | Authenticates users with a SQL Server-specific username and password. | Relies on Windows OS credentials for authentication. |
| **Credentials Storage** | Credentials are stored in the SQL Server database. | Credentials are managed by the Windows operating system. |
| **Security** | Depends on password strength and SQL Server security configuration. | Offers integrated security through Kerberos or NTLM protocols. |
| **Management** | Requires managing separate SQL logins and passwords. | Simplifies user management with existing Windows accounts. |
| **Use Cases** | Suitable for non-Windows environments or external applications. | Preferred for intranet applications or when using Windows domains. |
| **Connection Setup** | Requires explicitly specifying the username and password in the connection string. | Uses the current Windows user credentials automatically. |
| **Ease of Use** | Requires additional setup for creating and maintaining logins. | Simplifies setup with no need for separate user credentials. |
| **Syntax Example** | Connection string: User ID=sqluser;Password=sqlpassword;. | Connection string: Integrated Security=True;. |

**Inline Function and View**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Inline Function** | **View** |
| **Definition** | A SQL function that returns a table and allows encapsulating a single SELECT statement. | A database object that presents data through a predefined SQL query. |
| **Purpose** | Used for reusable logic involving a single SELECT statement. | Provides a virtual table for simplifying complex queries or joining data. |
| **Execution** | Requires invocation via SELECT or as part of a query. | Automatically executed when queried, like a regular table. |
| **Complexity** | Limited to a single SQL statement. | Can encapsulate complex queries but does not allow procedural logic. |
| **Modifiability** | Cannot modify underlying data. | Cannot directly modify data but can be used in INSERT, UPDATE, or DELETE operations if updatable. |
| **Performance** | Generally faster due to inline execution plans. | May have additional overhead depending on query complexity and database optimization. |
| **Use Cases** | Simple calculations, filtering, or projections. | Creating reusable query objects or simplifying access to complex joins and aggregations. |
| **Syntax Example** | CREATE FUNCTION FuncName() RETURNS TABLE AS RETURN (SELECT ...); | CREATE VIEW ViewName AS SELECT ...; |

**Identity and Unique Constraint**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Identity** | **Unique Constraint** |
| **Definition** | A property that generates unique values for a column, typically for primary keys. | A constraint ensuring that all values in a column or combination of columns are distinct. |
| **Purpose** | Automatically generates sequential numbers for new rows. | Prevents duplicate entries in a column or group of columns. |
| **Enforcement** | Enforced by the database engine when inserting new rows. | Enforced by database rules on data insertion or updates. |
| **Scope** | Limited to a single column, usually numeric. | Can be applied to single or multiple columns. |
| **Usability** | Typically used for primary key generation. | Used for ensuring data uniqueness in non-primary key columns. |
| **Value Specification** | Values are generated by the database and cannot be explicitly inserted (unless SET IDENTITY\_INSERT is used). | Values must be explicitly provided by the user and adhere to the uniqueness rule. |
| **Syntax Example** | CREATE TABLE TableName (ID INT IDENTITY(1,1), ColumnName VARCHAR(50)); | CREATE TABLE TableName (ColumnName VARCHAR(50) UNIQUE); |