**SQLServer**

**Lab 9**

**Name : Marwan Khalil Mahmoud Shamso**

What is the difference between the following objects in SQL Server :

1. **batch, script and transaction**

* A **batch** is a collection of Related or unrelated SQL commands that are executed together, They do not guarantee success or failure as a whole; each statement within a batch is executed independently, and Batch can execute when a certain condition is met.
* A **transaction** is a collection of SQL commands that are treated as an atomic unit of work to ensure atomicity and consistency, Transactions guarantee either complete success or complete failure If any part of a transaction fails, all changes made by the transaction are undone
* An **SQL script** is a file containing a sequence of SQL commands or statements Executed Sequentially, That used to Automate Tasks or define database objects.

1. **trigger and stored procedure**

* **Stored Procedure** Is A Block of Batch Code That Optionally Take Parameters  
  and Execution of the stored procedure Using Execute Command
* **Trigger** Is A Block of Batch Code doesn't Take Parameters  
  and Execution of the Trigger done automatically Instead of or After (Insert , update , delete) Commands

1. **stored procedure and functions**

* **Stored Procedure** Is A Block of Batch Code That Optionally Take Input and Output Parameters, Return Value is Optional and Execution of the stored procedure Using Execute Command.
* **Function** Is A Block of Batch Code That Take Only Input Parameters, Return a Single Value and Execution of the stored procedure Using Select , where and having Commands.

1. **drop, truncate and delete statement**

* **Drop** is (DDL Command) used to Permanently Remove Entire Table
* **Delete** is (DML Command) used to remove specific Rows from a table based on the conditions (after where clause)
* **Truncate** (DDL command) used to delete All Rows from a table , Faster than DELETE since it truncates all rows as a block and can't include Where clause

1. **select and select into statement**

* **Select** is used to retrieve data from one or more tables or expressions
* **Select Into** is used to Creates a new table based on the result of a query.

1. **local and global variables**

* **local variable** is declared within a specific block of code Starts with @, Not visible outside the block and lost when the block terminates.
* The system maintains global variables (system functions) starts with @@, used for system-related data , Cannot be declared by users and Store session-related information.

1. **convert and cast statements**

* **CAST** is Part of the ANSI-SQL specification, Converts data from one data type to another. Syntax: CAST(expression AS target\_data\_type), No additional formatting options and Widely supported across different databases.
* **CONVERT** is SQL Server-specific function, Also converts data types, Syntax: CONVERT(target\_data\_type, expression [, style]), Allows an optional style parameter for formatting (e.g., date formats) and More flexible for specific output requirements.

1. **DDL,DML,DCL,DQL and TCL**

* **DDL (Data Definition Language)** deals with defining and modifying the database schema.  
  Examples:
  + - CREATE TABLE: Creates a new table.
    - ALTER TABLE: Modifies the structure of an existing table.
    - DROP TABLE: Deletes a table.
    - TRUNCATE TABLE: Removes all records from a table.
    - COMMENT: Adds comments to the data dictionary.
    - RENAME: Renames an object in the database.
* **DML (Data Manipulation Language)** is used for manipulating data within the database.  
  Examples:
* INSERT INTO: Adds new records to a table.
* UPDATE: Modifies existing records.
* DELETE FROM: Removes specific records.
* SELECT: Retrieves data from tables.
* **DCL (Data Control Language)** manages access and permissions.  
  Examples:
  + - GRANT: Provides specific privileges to users.
    - REVOKE: Removes privileges.
    - Deny : deny specific privileges to users
* **TCL (Transaction Control Language)** deals with transactions.  
  Examples:
  + - COMMIT: Saves changes made during a transaction.
    - ROLLBACK: Undoes changes and restores the database to a previous state.
    - SAVEPOINT: Sets a point within a transaction for later rollback.
    - SET TRANSACTION: Configures transaction properties.
* **DQL (Data Query Language)** is used for querying data from schema objects.  
  The primary command is SELECT, which retrieves data based on specified conditions.

1. **For xml raw and for xml auto**

* **For XML RAW** converts the table to XML File contain the data of the Rows in the table into XML elements with tags , can include aggregated columns and GROUP BY clauses
* **For XML Auto** Returns query results in a simple, nested XML tree. Each row returned by the query is represented by an XML element with the same name. The columns listed in the SELECT clause are mapped to the appropriate element attributes.

1. **Table valued and multi statement function**

* **Table valued** is used with function and Stored procedure to pass a table as a single parameter to Function or SP it can be used to pass data from SP to another, it is read only so it can't be treated output parameter in SP, variables that is created from Type TVP is saved in Tempdb
* **multi statement function** Same as Table Valued But it can return a user-defined-structured table as Output

1. **Varchar(50) and varchar(max)**

* **Varchar(50)** specified Fixed Length for character data, stores up to 50 characters
* **Varchar(Max)** Stores variable-length character data up to (2 GB)characters **,** Dynamically Allocated spaces based on the actual data Length.

1. **Datetime(3), datetime2(7) and datetimeoffset(7)**

* **Datetime(3)** Represents date and time values with a fixed precision of 3 milliseconds, It's Storage Size is 8 Bytes and doesn't include time zone information.
* **Datetime2(7)** Represents date and time values with Higher Precision than DATETIME, Allows Fractional seconds up to 7 decimal places , Storage size varies based on Precision and doesn't include time zone Information**.**
* **datetimeoffset(7)** Stores date and time values along with an offset from UTC, Precision: Up to 7 decimal places for fractional seconds and Storage size varies based on precision and time zone offset.

1. **Default instance and named instance**

* **A default instance** is used when installing a single instance of SQL Server.

It has no name; when you provide only the computer name, the connection is made to the default instance**.**

* **A named instance** is used when you specify an instance name during installation.

You connect to a named instance by providing both the computer name and the instance name.

1. **SQL and windows Authentication**

* **SQL Server Authentication** : Logins are created in SQL Server independent of Windows user accounts. Both the username and password are managed within SQL Server. Not as secure as Windows Authentication.
* **windows Authentication** : Users log in using their Windows user accounts. SQL Server validates the account name and password using the Windows principal token, No password is required during connection; Windows confirms the user’s identity.

1. **Clustered and non-clustered index**

* **The clustered index** physically Order and stores data rows in the table based on their key values. It defines the order in which records are stored on disk. There can be only one clustered index per table because the data rows themselves can be stored in only one order. The primary key of a table is used as the clustered index.
* **The Non-clustered index** has a separate structure from the data rows. It contains the non-clustered index key values and pointers to the data rows. The pointer from an index row to a data row is called a row locator. Non-clustered indexes allow multiple indexes per table. You can add non-key columns to the leaf level of a non-clustered index for fully covered queries.

1. **Group by rollup and group by cube**

* **The ROLLUP** **operator** extends the functionality of GROUP BY by calculating subtotals and grand totals for a set of columns. It generates additional rows that represent subtotals and grand totals. The order of columns specified in the ROLLUP determines the hierarchy of subtotals.
* **The CUBE operator** is similar to ROLLUP, but it calculates subtotals and grand totals for all permutations of the specified columns. It produces output rows for each unique combination of values across the specified column. Additionally, CUBE includes individual values for each column, not just subtotals.

1. **Sequence object and identity**

* **Sequence Object** is created by the user and is independent of any specific table.   
  It can be shared across multiple tables.
* **The IDENTITY property** is associated with a specific table column. It generates an auto-incrementing numeric value for that column. Since it’s a table-specific property, it cannot be shared among multiple tables.

1. **Inline function and view**

* **View** can't take parameters, is A select statement , it Simplify construction of complex queries , Specify user view , You can Limit access to data [grant revoke, Hide names of database objects )table name and columns)  
  types of views :  
  - Standard view  
  - partitioned view  
   -indexed views
* **Inline function**  is a parameterized view that can be expanded and optimized away just like a view. It returns a table-like result set based on the input parameters.

1. **Table variable and Temporary table**

* **Table** variable Created in memory (RAM. Faster for small to medium volumes of data. SELECTs are implicitly with NOLOCK. Useful for encapsulating logic within functions and reusing it. Not recompiled during stored procedure execution. Dropped once out of the batch (e.g., when a procedure finishes).
* **Temporary table** Created in the tempdb database. Real tables, allowing operations like creating indexes. Ideal for large data volumes where indexed access is faster. Participate in transactions and locking. May result in stored procedures being recompiled. Can be created using SELECT INTO for ad-hoc querying.

1. **Row\_number() and dense\_Rank() function**

* **ROW\_NUMBER()** Assigns a unique, incremental number to each row within the result set. No ties: Each row receives a distinct number. Useful for pagination, ranking, and identifying individual rows.
* **DENSE\_RANK()** Handles ties by assigning the same rank to equal values but doesn’t skip ranks Tied rows get the same rank, and the next rank continues without gaps