1.

Batch:

- it is a collection of SQL statements that are grouped together and executed as **a single unit**.
- can include a combination of DML statements (INSERT, UPDATE, DELETE) and DDL statements (CREATE, ALTER, DROP).
- Statements inside it don’t affected by each other, if one statement is failed, the others execute عادي.

Script:

- It is a file or sequence of SQL statements that are executed in a **sequential manner**, separated by GO statement. It can contain multiple batches or a single batch.
- You can create a script for database and run it in any other server or machine to create the same database with all views, constrains, sp,…

Transaction:

- It ensures that either all the operations within the transaction body are successfully executed(then commite), or none of them are applied(then rollback) - known as the "all-or-nothing" principle.
- In a transaction, you typically begin with a BEGIN TRANSACTION statement, perform your database operations (such as INSERT, UPDATE, DELETE), and then either commit the changes using the COMMIT statement or roll back the entire transaction using the ROLLBACK statement if any issues or errors occurred. (ACID).

2.

Stored Procedure:

- It is like a repo that contains some SQL statements and it is executed when u invoke it using application code or database tools. And I can write all types of queries (DML, DDL, If conditions, Loops, joins, execute dynamic queries …)
- It provides high security, because the names of objects, type of query executed, metadata, all of that are hidden. (prevent SQL Injection).
- It provides high performance for the network, because it decrease the number of characters on the network as u call the name of the proc only, and high performance for the engine as the cycle of executing query(parse -> optimize -> query tree -> execution plan) is performed just at the first call of the proc, and stored in memory.

Trigger:

- it is a database object that is created on the table and is automatically fires when a specific event (such as an insert, update, or delete) occurs on that table. They are used to enforce data integrity rules, perform auditing, etc.
- Triggers do not accept input parameters. However, they can access the data of the modified rows الصفوف اللي عملتلها تعديل في الجدول through a "virtual tables" available within the trigger code --> "**inserted**" and "**deleted**" tables.
- Triggers fires either the query is effect on the table(the trigger is created on) or not.
- Trigger take the same name schema of the table it is created on.
- U can enable or disable the trigger from the table. U can drop it too.

3.

Function:

- It always returns a value., the return value can be a single value, a table, or a result set.
- They cannot have output parameters, input ones only.
- They can be used as a part of SQL statements, such as SELECT statements, WHERE clauses, or JOIN conditions. It is like a subquery.
- Cannot handle the errors using try catch inside functions.

Stored Procedure:

- It may or may not return a value.
- Can have output parameters an input parameters.
- Cannot be used directly in SQL statements.
- Can use any type of handling errors inside SP.

4.

DROP:

- Remove database objects such as tables, views, indexes, stored procedures, rules, constrains, …etc. It permanently deletes the object and its metadata.
- It cannot be rolled back.

TRUNCATE:

- Remove all rows from a table while keeping the table structure.
- Cannot use WHERE statement with it.
- Fast alternative to the DELETE command when you want to remove all the data from a table as it is not save in the log fileغالباً.
- Resets any auto-incrementing primary key values(Identity).
- It cannot be rolled back if it is not within transaction scope, if it is and the transaction failed it will be rolled back.

Delete:

- Remove specific rows based on condition.
- Can use WHERE statement with it.
- Slow because it is saved in the log file.
- Do not reset Identities.

5.

Select:

- Use to query and retrieve data from one or more tables.

Select Into:

- Use to create a new table and insert data inside it these data may be came from another select statement, views, functions, or another table(physical or variable in memory).

6.

Local Variables:

- Have a limited scope and are only accessible within the specific block(batch) or procedure where they are defined.
- They are created in memory when the batch starts executing and destroyed when it completes.

Global Variables:

- Have a wide scope and can be accessed from different parts of the database system.
- Used to store values that need to be shared across different parts of the database system.

7.

Convert:

- CONVERT (data_type, expression, [style]).
- More flexible than CAST.
- Can handle some conversion errors by returning NULL or a default value specified by the ISNULL() function.

Cast:

- CAST(expression AS data_type).
- Does not provide built-in error handling. If a conversion cannot be performed successfully, it will result in a runtime error.

8.

DDL:

- It is a group of queries used to <u>manage</u> the **structure** of database objects such as tables, views.
  (CREATE, ALTER, DROP, TRUNCATE)

DML:

- It is a group of queries used to <u>manipulate</u> or <u>change</u> the data in tables or views.
  (SELECT, INSERT, UPDATE, DELETE)

DQL:

- It is a group of queries used to <u>retrieve</u> data from one or more database objects.
  (SELECT)

TCL:

- It is a group of queries used to <u>manage</u> **transactions** in a database
  (COMMIT, ROLLBACK)

9.

For xml raw:

- Convert the physical table in database to an XML tags and attributes. It could not distinguish join statement.

For xml auto:

- Convert the physical table in database to an XML tags and attributes. It could distinguish join statement.

10.

Table-Valued Function:

- It is a user defined function that returns a table as an output.
- Could be used in a SQL statements.
- It is created by CREATE FUNCTION and returns table.
- Has no function body.

Multi-Valued Function:

- It is a user defined function that can contain multiple SQL queries and any logic, and returns a TABLE.
- Has a function body
- It contains a table variable type and we have to write the schema name when call it.
- Have body to inserts data into this variable.

11.

Varchar (50)

- Varchar (50) can store maximum of 50 characters. If the string stored inside it more than 50 characters, it will be cut to fit the 50 cahrs.
- It's more efficient because we predefine the amount of space needed for the variable.

Varchar(max)

- Varchar (max) can store the maximum amount of characters.
- It is less efficient as it uses variable amounts of storage space and memory.

12.

Datetime:

- It has a precision of 3.33 milliseconds, meaning it can store values rounded to increments of 0.00333 seconds.
- 8 bytes of storage.
- Format: YYYY - MM - DD HH:MM:SS

DATETIME2:

- It is an extension of the “datetime” type with an extended range and higher precision. It can store date and time values with a precision of up to 100 nanoseconds. It is used for more accurate representation of time values and greater flexibility in storing fractional seconds.
- The storage size varies depending on the precision specified. It ranges from 6 to 8 bytes.
- Format: YYYY - MM - DD HH:MM:SS.[nnnnnn]

DATETIMEOFFSET:

- It is used to store date and time values, including the time zone offset.
- 10 bytes of storage.
- Format: YYYY - MM - DD HH:MM:SS[.nnnnnn] {+|-}HH:MI

13.

Default Instance:

- Use the default name of the machine and associated with the server's hostname or IP address.
- When connecting to a default instance, we don't need to specify the instance name, just <u>ServerName</u> or <u>Dot</u>.

Named Instance:

- Have unique name and password.
- When connecting to a named instance, we must specify the instance name in the connection string, <u>ServerName\InstanceName</u>.

14.

SQL Authentication:

- Accessing the server of the database using a SQL username and password.
- It is used mostly when the server of database is an online server(have space host محجوزاله on the internet), because this server don't know the name of each machine will connect to it, so give a unifiedمُوَحد username and password, and all users connect to this server using it.

Windows Authentication:

- Accessing the server of the database using your windows username and password.
- It is used mostly when the server of database is on a server machine in a shared internal networkشبكة داخلية and all computers can access this server. In this case, each computer can access the database using its own windows username and password. If one of the computers use Linux, then he cannot access using windows authentication, he have to access using SQL username and pass.

15.

Clustered index:

- There is only one clustered index in the table.
- It creates a binary-tree in memory that has the actual(physical) data pages as the leaf nodes level that are ordered according to the index key, so it is very fast.

Non-clustered index

- We can have more than one non-clustered index in the table.
- Nonclustered indexes have the same binary tree structure as clustered indexes, but it takes a copy of the physical data pages, then order them, and make each copy value points to the similar values in the physical pages →(the data and the index are stored separately), so it slow compare to Clustered.

16.

GROUP BY *ROLLUP*:

- It generates summary rows for each combination of values in the grouped columns, plus a total summary row for all the grouped columns.

```
select SalesmanName, sum(quantity) as Qty
from sales
group by rollup(SalesmanName)
```

  This query will return the sum of quantities of each product for each Sales man, plus another row at the end represents the sum of all quantities of all sold products for all sales men.

GROUP BY *CUBE*:

- It generates summary rows for each combination of values in all the grouped columns, including each individual column and combinations of columns.

```
select ProductID, SalesmanName, sum(quantity) as "Quantities"
from sales
group by cube(ProductID, SalesmanName)
```

This query will add:

Sum of quantities of all products for each salesman(3 rows).

Sum of the quantity for each product sold by all salesmen(4 rows).

Sum of all quantities of all sold products for all sales men(1 row).

17.

Sequence:

- It generates a sequence of unique numeric values based on a specified starting, increment and max value.
- It can be used by multiple columns and tables.
- Have a cycle option.

Identity:

- It generates a sequence of unique numeric values for a single column in a table.
- It could be controlled too using ON or OFF option, if u have a gap between identities due to delete statement.

18.

Inline function:

- Can have parameters.
- returns a table datatype, and the name of the function can be used as a table in another query.
- Its body contains only select statements because it returns table, and u cannot make other DML or DDL queries on it.

View:

- Cannot have parameters.
- Returns nothing, and the view can be used in quires as a table too.
- It is a virtual tables, can be considered a layer of security to hide metadata.
- Its body contains only select statements, no DML or DDL inside it, but u can make Update, insert, delete **on** the view based on some rules.

19.

Table Variable(@t table):

- It is limited to the scope of the current batch, stored procedure, or function, so when the scope ends of execution the table is destroyed.
- It is created using DECLARE statement and it is stored in memory.
- It is not saved in log file, meaning it cannot be rolled back.

Temporary Table(# or ##):

- It is visible to all sessions and users(wide scope).
- It is created using CREATE TABLE #table_name, and is stored in **tempdb** database that exists in “System Databases” file as a physical table.
- It’s lifetime depends on the instance(user), it is destroyed if the server is shutdown or the user ends the session.

20.

Row_number() Function:

- It assigns a unique sequential number to each row in the result set, regardless of duplicate values in the ordering columns. 1,2,3,4,5,6,7, …

Dense_Rank() Function:

- provides a rank to each row based on the specified ordering, where rows with the same values receive the same rank.

```sql
select * , Dense_Rank() over(order by st_age desc) as DR
from sc.student
```

- This query will rank the rows based on ages, all same ages take the same rank 1,1,1, 2,2 ,3,3, 4,4,4,4, 5, 6,6 , …