**SQL Statements:**

**DDL Statements (Data Definition Language):**

1. CREATE, DROP, ALTER, TRUNCATE, RENAME

**DML Statements (Data Manipulation Language):**

1. SELECT, INSERT, UPDATE, DELETE

**TCL Statements (Transaction Control Language):**

1. COMMIT, ROLLBACK, SAVE TRANSACTION

**DCL Statements (Data Control Language):**

1. GRANT, REVOKE

* Database is the collection of Tables, Views and Triggers.
* WHERE Clause is generally used with three commands: “SELECT”, “UPDATE” and “DELETE”.
* General Aggregate functions or group functions used in SQL Server: SUM(), AVG(), MIN(), MAX(), COUNT()
* General Numeric functions are ABS(), CEILING(), FLOOR(), SIGN(), SQUARE(), SQRT()
* General String functions are LEN(), UPPER(), LOWER(), SUBSTRING(), LTRIM(), RTRIM().
* Difference between Where and Having clause is that Where clause can be used with DML statements independently while Having Clause is dependent on Group By clause and will always use with Group by Clause.
* TOP clause is always used with “Order By” Clause and it is used in the start right after SELECT statement.
* INTO Clause is used to copy the data (specific columns or complete Table) from another DB or from the same DB as well.
* While Altering the Table and changing the size of any column, do not decrease the size beyond the length of the data, which is already present in the Table.
* JOINS are used to retrieve data from multiple Tables.
* 3 Types of JOINS:
  + Inner Join / Simple Join / Natural Join (It returns all the rows from multiple tables where Join condition is satisfied).
  + Outer Join (Left Outer join, Right Outer join, Full Outer join)
  + Cross join
* In order to perform Join between Tables, they must have at least one column data similar in between them.
* In Left Outer JOIN, Right Outer Join and Full Outer Join records without matching values are replaced with NULLs in the respective columns.

**Subquery:**

* A query within another SQL query and embedded within Where clause.
* Subquery must be enclosed within parenthesis ().
* Subqueries can be used with SELECT, UPDATE, INSERT and DELETE statements along with the comparison operators.
* A subquery can have only one column in the Select Statement.

**Transact- SQL:**

* The purpose of T-SQL is used to provide a set of tools for the development of Transactional database.
* **Why T-SQL:** Because it includes Standard SQL, Working with variables, Batch or Script, User defined functions, Triggers, Stored Procedures and many more….

**Working with Variables:**

* Variables are declared in T-SQL with the command “DECLARE” followed by variable name preceded with @ symbol and data type. For example:
  + DECLARE @name VARCHAR (50);
  + Declare @name VARCHAR (50), @Age INT;
* By default, the value of declared variable is NULL.
* Both SET and SELECT command can assign the value to a variable.
* SET command is used to assign value to a single variable at a time.
  + SET @name = ‘Adil’;
* SELECT command is used to retrieve data from tables and assign values to multiple variables in a single statement using comma separators.
  + SELECT @name = ‘Harry’, @Age = 25;
* **Mathematical operations on Variables:** SET @number += 10; or SET @number = @number+10; and SET @number ­-=10; or SET @number = @number -10; and SET @number \*=10; or SET @number = @number \*10;
* A batch will always end at GO statement following single or group of statements in a script file and a script file consists of lots of batch statements.

**Batch:**

* A batch of SQL statements is a group of two or more SQL statements or single SQL statement.
* A batch can include the following statements :
  + DDL
  + DML
  + DCL
* Standard type of Batches:
  + Explicit Batch: Two or more SQL statements separated by semi-colon (;).
  + Procedure: if a procedure contains more than one SQL statements, then it is considered to be a batch.

**GO:**

* Go is not a T-SQL statement. It is a command recognized by SQL server utilities.
* It signals the end of the batch to SQL Server utilities.
* Go can be executed by any user. It requires no permission.
* Syntax Go [count]
  + Where count is a positive integer, which tells the SQL server utilities the number of times, this batch will run. Default value of count is 1.
* Variables declared in a batch are local variables which are only applicable till the GO statement and they will not be applicable in the next batch.

**Control of Flow:**

* Transact SQL statements are executed in sequential order but sometimes we want to interrupt the normal flow of execution.
* T-SQL has keywords to control the flow of execution. In T-SQL, these are known as control-of-flow. These key words are as follows:
  + BEGIN…END, If…Else, While, Break, Continue, Goto, Return, Try…Catch, Throw, WAITFOR

**BEGIN…END:**

* The BEGIN…END keywords are used to group multiple lines into one statement block.
* BEGIN…END can be nested, meaning we can place a BEGIN…END statement within another BEGIN...END statement.
* BEGIN...END will be used every time while working in T-SQL.

**IF…Else:**

* If the condition used with “if” keyword is of select statement then it will always be enclosed within parenthesis.

**Try…Cath:**

* TRY…CATCH implements error handling for T-SQL.
* A group of T-SQL statements can be enclosed in a TRY block.
* To retrieve the information about the error in catch block, here are some pre-defined functions:
  + ErrorMessage() => Returns the complete text of the error message.
  + ErrorNumber() => Returns the error number of the error.
  + ErrorLine() => Returns the line number in the routine which caused the error.
  + ErrorProcedure() => Returns the name of the stored procedure or trigger where the error occurred.
  + ErrorSeverity() => Returns the severity of the error.
  + ErrorState() => Returns the error state number.

**WAITFOR:**

* WAITFOR blocks the execution of a batch, stored procedure or transaction until either a specified time or time interval elapses or a specified statement modifies or returns at least one row. WAITFOR has two arguments:
  + Time => the period to wait. Time\_to\_pass
  + Delay => the time (up to maximum 24 hours) at which the WAITFOR statement finishes.
* WAITFOR TIME ‘time to execute’, WAITFOR DELAY ‘time to pass’ => Both ‘time to execute’ and ‘time to pass’ can be specified either in date time data format or as a local variable.

**Stored Procedure:**

* A stored procedure is a group of one or more T-SQL statements.
* It can be stored in database.
* It accept input parameters and return multiple values.
* Contain programming statements that perform operations in the database.
* Return the status value to a calling program to indicate its success or failure.
* **Benefits =>** Reuse of code, Improve performance, strong security, easy to maintain, reduce client/server network traffic.
* **Types of stored procedure:**
  + **System:** physically stored in the internal resource database.
  + **User-defined:** It can be created in a user-defined database or any system database except resource database.
  + **Temporary:** A form of user-defined procedures are like permanent procedure except they are stored in Tempdb. Its scope is very limited and are not created to use for long-term purpose.
* Stored procedure cannot be called inside clause like “Where” clause.
* Two ways to create (define) a stored procedure:
  + Stored Procedure without parameters (simple stored procedure)
  + Stored Procedure with Parameters
* **Rename a stored procedure:**
  + We need system procedure to rename an existing procedure.
  + “Sp\_rename” is a system procedure to rename an existing procedure.
  + Do not use “sp\_” keyword at the time of defining a procedure, as it is a reserved keyword for system procedure.
  + Drawbacks of renaming:
    - Renaming a stored procedure does not change the name of the corresponding object name in the definition column of sys.sql\_modules catalog view. To do that we must drop the stored procedure and recreate it.
    - Changing the name or definition of stored procedure can cause dependent objects to fail when objects are not updated to reflect the changes that have been made to the procedure.

**User-defined functions (UDFs):**

* It return either a single scalar value or a result set.
* Function can be called in clause like “Where”.
* **WHY UFDs:**
  + Every time an application runs, it does not require to reparsed and reoptimized function. It only does it once. So saving time and resources. (Reduce the compilation cost of Transact SQL).
  + Function can be invoked in where clause to reduce the number of rows sent to the client.
* **Types of Function:**
  + System: They cannot be modified.
  + Scalar: Return a single data value of the type defined in Return clause.
  + Table-valued: return a Table data Type.
* **Before creating a function, things to know:**
  + User-defined function always return a value.
  + User-defined function only have input parameters for it.
  + User-defined function cannot return multiple result sets.
  + A UDF cannot use SET statement in it as it is not allowed.
  + UDF cannot call stored procedure but a stored procedure can always call user-defined function.
  + Error handling is restricted in User-defined function i.e. it does not support TRY…CATCH, @ERROR, RAISERROR.
  + User-defined function can be nested. AUDF can be nested up to 32 levels.
* **Table-Valued function:**
  + Returns a value of table data type.
  + It can accept zero or more parameters.
  + Types of Table-valued function:
    - Inline Table-valued function: There is no function body and does not require a BEGIN…END block because BEGIN…END applies on multiple statements block.
    - Multi-statement table-valued function:
      * It consists of multiple SQL statements and enclosed in BEGIN-END block.
      * We have to declare a variable in RETURNS statement with Table data type and return statement is without a value and function will return that declared table variable automatically

**Trigger:**

* A Trigger is a type of stored procedure that automatically runs when an event occurs in the database server.
* Events are DML operations (INSERT, UPDATE, and DELETE).
  + **Types of Triggers:**
    - **DDL Triggers:** 
      * DDL Triggers are fired in response to different DDL events on DDL statements (CREATE, ALTER, DROP, GRANT and REVOKE). These are always after tiggers.
      * In addition, some system stored procedures, which perform DDL like operation like sp\_rename can also fire DDL trigger.
      * DDL Triggers can be performed on either DATABASE or SERVER. (both are keywords)
    - **DML Triggers:** DML Triggers are fired on DML statement (INSERT, UPDATE AND DELETE) and stored procedures. DML Triggers are of three types.
      * **Types of DML Triggers:**
        + **After Triggers:** These triggers executes after the action of INSERT, UPDATE, MERGE AND DELETE statement is performed.
        + **Instead of Triggers:**

These triggers overrides the standard actions of the triggering statement.

It can be used to perform error or value checking on one or more columns.

These triggers perform additional actions before insert, update and deleting the row or rows.

* + - * + **CLR Triggers: (Common Language Runtime Trigger)**

It can be either after or Instead of Triggers.

It can also be a DDL Trigger.

**Magic Tables:**

* SQL Server automatically creates and manages magic tables. DML Trigger statements use two magic tables.
  + **Inserted Table**: This table stores the copies of the affected rows during INSERT and UPDATE statement. During these transactions, new rows are added to both the Trigger Table and the Inserted Table. Inserted table contains the rows after the INSERT and UPDATE operation is performed (the new rows).
  + **Deleted Table:** This table stores the copies of the affected rows during UPDATE and DELETE statement. During these transactions, rows are deleted from the trigger table and stored into the Deleted table. Delete Table contains rows before the UPDATE and DELETE operation is performed (the old rows).

**WHY DDL Triggers:**

* Prevent certain changes to your database schema.
* Have something occur in the database in response to a change in your database schema.
* Record changes or events in the database schema.

**EVENTDATA():**

* It is a built-in function.
* It returns the information about the events executed by DDL trigger.
* It returns the information in XML format.

**Merge:**

* Merge is a logical combination of an INSERT and UPDATE.
* It combines the sequence of conditional INSERT, UPDATE and DELETE statements into single SQL statement.
* Using merge statement, you can sync two different tables so that content of target table is modified based upon the differences found in the source table.

**Why Merge:**

* It is specially used to maintain history of data in data warehousing during the ETL (Extract, Transform, Load) cycle.
* **Scenario**: Suppose tables need to be refreshed periodically with new data arriving from an online transaction processing (OLTP) system. This new data may contain changes to existing rows in tables and/or new rows that need to be inserted.

**Action/Condition for the Merge Statement:**

* If rows in the source table are not found in target table then rows will be INSERT to target table.
* If rows which are present in target table but are not found in the source table. Those rows will be deleted from target table.
* Rows in the target and source tables have the same keys but they have different values in non-key columns then data of target table will be updated from data of source table.

**States of MERGE condition:**

* **MATCHED:** These are the rows that match the merge condition. For the matching rows, you need to update the row columns in target table with values from source table.
* **NOT MATCHED BY TARGET:** These are the rows that are present in the source table but doesn’t match with the target table so those rows will be added from source table to target table.
* **NOT MATCHED BY SOURCE:** These are the rows, which are present in the target table but does not match with the source table. If you want to synchronize the target table with the data from the source table then those rows will be deleted from target table.
* It’s not necessary to use all three conditions in one statement. We can use one or two conditions as well but it needs to be terminated by a semi-colon.

**Index:**

* Databases systems uses indexes to provide fast access to relational data.
* It is a special type of physical data structure used to access one or more data rows fast.
* **Database index can change every time the corresponding data is changed.**
* The system component called the query optimizer decides whether to use an existing index or not.
* Indexing is used to arrange the data in either ascending or descending order before sending the result back to the user.
* Data constraint “PRIMARY KEY” by default creates an Index for the data values to be inserted in the database. “PRIMARY KEY” can be used on id column or any other column of our choice.

**Clustered Index:**

* A clustered index determines the physical order of the data in the table; hence a table can have only one clustered index.
* When a clustered index is created, database engine sorts the data based upon the defined Index key(s) and stores the table in that order.
* A primary key constraint creates clustered index automatically if there is no clustered index exists on the table already.
* An index can contain more than one column known as composite index.

**Non-Clustered Index:**

* A non-clustered index does not change the physical order of the rows of the table.
* In simple words, in non-clustered index, data is stored in one place and index is in another place. The index will have pointer to the storage location of the data.
* Since index is stored separately from the actual data, a table can have more than one non-clustered index.
* It is faster in searching for values that are not in range.

**UNIQUE Index:**

* A unique index ensures that there are no duplicate values in Index key.
* There is no difference between creating a UNIQUE constraint and creating a UNIQUE index.
* UNIQUE is a property of clustered and non-clustered index.

**Implementation of UNIQUE Index:**

1. **PRIMARY KEY:** when you create a Primary key constraint, a unique clustered index on column or columns is automatically created.
2. **UNIQUE constraint:** when you create a unique constraint, a unique non-clustered index is created to enforce unique constraint by default.
3. **Index Independent of a constraint:** Multiple unique non-clustered indexes can be defined on a table.