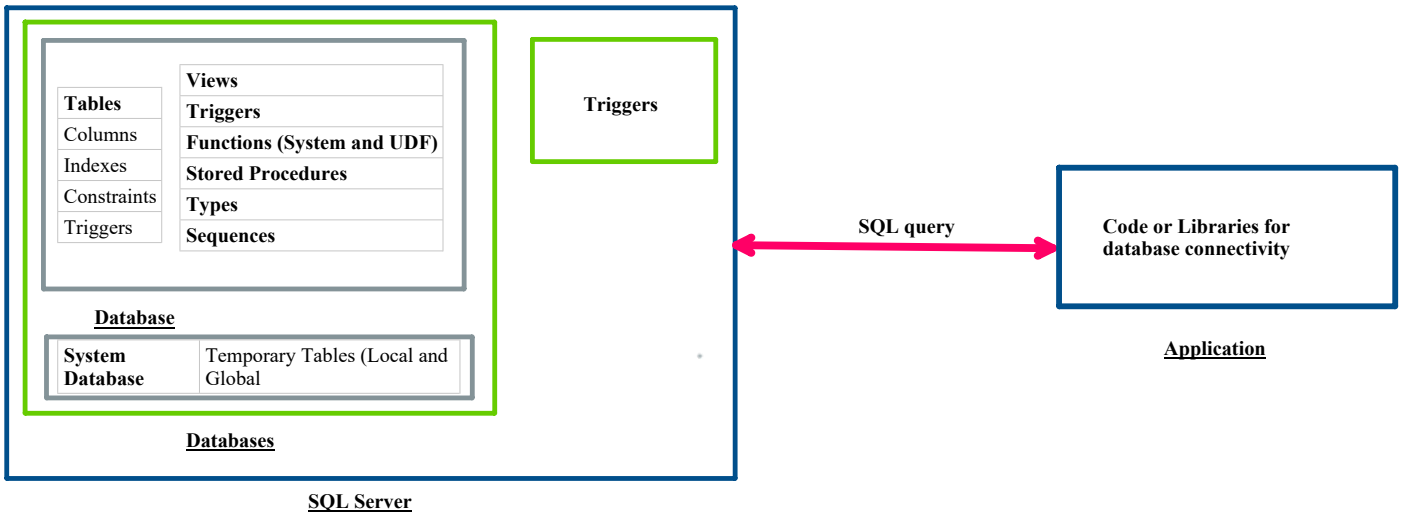
SQL Server Tree

Thursday, February 18, 2021 9:37 AM



1. **First Main Storage Things are**
   - **Temporary Tables**
   - **Tables (Permanent)**

A. Tables or Permanent Tables
- Can Store data as long as we wish
- Can Add Constraints on column Like
  - Default, Unique, Check
  - Primary Key, Foreign Key
  - Index
- Can Create DML Triggers and also DDL Triggers are also
- Followed By Tables.

A. Temporary tables

a. Local:
- Which Individual for Every User
- As Connection By User is Close table got destroy
- As Individual For User so possible Same Tables by Different Users so some Value is There along with table name, Which help to store duplicate table uniquely and stop same user to create duplicate table

b. Global
- Which Global among Every Connection
- No Duplication Allowed as All User can Access once Created By Any User
- Get Terminate when last user who using that terminate the connection, No matter who created that.
-

- Note : DDL Trigger Not Applicable, View and Function is Not Applicable or temporary function, if temporary table is in store proc or function then as that function complete they got destroy.
- Other then that we can have all options which we can implement using table like constraint, Index, Keys and Many More.
- While Creating Table Just have to Add Prefix in table name # for Local and ## for Global

2. **Triggers**
   - Mainly Used Triggers are
     - **DML :** On Particular Table To Mange Manipulation of data Events
       - There are Two Types of DML triggers are there FOR and Instead of
       - FOR: Trigger after Operation P{performed
       - INSTEAD: Instead of that event, Mainly Useful for Manipulation of data from View
       - **Signature:**

```
Create trigger <trigger name>
 on tblname
For / instead Insert/Update/Delete
As
Begin
      // Code
End
```

       - In This Inserted and Deleted Table is Useful.

     - **DDL : It is used on Database or Server to trigger on defining or altering or dropping particular definition.**
       - **Signature :**

```
Create trigger <trigger name>
 on database / ALL server
For Event
As
Begin
      // Code
End
```

        - **LOGON:** If Event in DDL is Logon then it is logon trigger, and code is related that and generally it is on server
            - To Get Login Details sys.dm_exec_sessions table is used
            - To Get Name of Current User Original_Login()

    - Can Set Order of Triggers using sp_settriggerorder including paras like @triggername, @order : first/last, @namespace : database or server, @stmttype: Can Say Event

3. **Data Types:**
    - There are many types of data types are available like int, float, bigint, decimal, varchar(size), varbinary(size) and many more
    - Date and Time Related Types are also there like
        - Time : 3-5 Bytes
        - Date : 3 Byte
        - SmallDateTime : 4 Byte
        - DateTime : 8 Byte
        - DateTime2 : 6 - 8 Byte
        - DateTimeOffset 8 - 10 Byte
        - Note: Just Differnec Between all is Precession like 1day, 1 min to 100ns, can store time zone too.
    - There are many inbuild functions are there which are available in to perfrom operations or Get data Like For Date GETDATE(),ISDATE() and Many More
    - DATALENGTH() is useful to get size of data type.
    - Sys.types table have information of all data types
    - UNIQUEIDENTIFIER or Guid is very helpful when we need to store data which helps to identify uniquely across server, No doubt as it is largest size of data storage have to use only when require, GETID() to get new GUID().
    - CAST and Convert Function Helps To Cast one data type to other if possible, cast is more preferable where as Convert is just helpful when we want to apply style on date specifically.
    - May Possible Value is Null and Want To Preplace then for That ISNULL and Coalescc is helpful function.
    - If We Want to pass table as argument then can create our table type to an then need to just create variable of that type and then just want to pas
        - Create type name as table
          (
              structure
)

4. **Constraint and Index:**
    - Basically Index is Useful to get data in very best way in terms of speed.
    - Types of Index are
        - Clustered, Non Clustered, Unique, Non Unique
    - Many Constraint like Primary key, Unique is using index indirectly.
    - Can Add Constraint Inline, At Time Of Declaration of table but after declaring all column or by altering table, Can also Drop the Constraint.
    - Few Constraint on Table where few are on column.
    - Can Add Identity to our column, and to check that Scope_Identity(), @@identity, Ident_current(tblname), To reset DBCC CHECKIDENT(tablname,RESEED,value).
    - Signature :
        - Index:
            - Create type/es index index name
              On tablename/viewname(column)
        - Constraint:
            - Constraint constraintname type(column name) if need then extra (On refrences tblname(column name) in fk)

5. **Basics of Tables:**
    - Can Create, Alter or Drop Table to deal with structure or definition of table : For Data Definition
    - Can Insert, Update, Delete or Truncate data of table : For data Manipulation
    - Can Select or Retrieve Data : For Data Retrieval

6. **Retrieval of Data:**
    - Can Retrieve data using Select column list query.
        - Simple Select Query With Column list.

    - Can Replace Null Values by Null Replacement or casting or function value

    - Can Join Table using Join
        - Different Types of Join To Join table on relation and then retrieval of data
        - Types of Join : Inner, left, right, full, cross

    - Can Operate Set Operation Like Union, Intersect, Minus(Except).
        - Similar like Join We Can operate Set Operation on two different set of data of same structure
        - Set Operations like Union, Union All, Intersect, Except

    - Can Filter/short Data, Group or Aggregate Data
        - Can Filter Data Based On Conditions where clause using Operators like IN, NOT, Between, LIKE, other Logical and Conditional Ops
        - We Can Group Data By Group by and Over clause

- Over clause is used with aggregate functions only and it can do partition and order by , Can Say Helpful for small grouping or instead of multiple set operations used like
  - Select SUM(colname) over(partition with colname) from tblname
- Group by is helpful to aggregate data with having to filter
  - Have GROUPING(colname) to check column is aggregated or not, Grouping_ID(col list) can say to check level
  - GROUP BY(col list), CUBE(col list), ROLLUP(collist),GROUPING set((col1,col2),(col1),(col2),())
- Order by is Useful to short data and when grouping is there then Grouping function is use here too for batter shorting, Distinct keyword to remove duplicates

- Can Insert Data retrieved data.
  - Select * into bactablename from table
  - To Copy Data or Structure of table. Note : Only Structure Not constraint

- More :
  - Case Statement : Like If else
  - Choose : Same like Switch
  - IIF : Like Ternary Operator.
  - Alias name: name as alias
  - Cross Apply and Outer Apply are like join but useful when join with table value function
  - We Can Store Particular Value it declare variable too instead of printing.

7. <u>Sub Query, Derived Table, Pivoting and Unpivoting:</u>

a. Sub Query: Generally to execute query into query is called Sub query.
b. Derived Table : Can Say it just Store Query or intermediate data just like view or temporary table to manage our execution ways lik
   i. Table variable
   ii. Select * from (select query) as tblname
   iii. CTE: with tblname (columnlist) as (select query).
c. Pivoting: Is Used To Rotate Data on any Dimension
   i. <u>Signature:</u>

```
Select Col1,col2
From tbl
PIVOT
(
    Agrregatefunction()
    For columnname IN (col1,col2)
)
```

Ex:

Gender	Population
Male	20000
Female	10244

Then PIVOTING ON GENDER

	Male	Female
Population	20000	10244

d. Unpivoting : Revert of Pivoting
   i. <u>Signature:</u>

```
Select Column,Value
From tbl
UNPIVOT
(
    Agrregatefunction()
    For columnname IN (col1,col2)
)
```

8. <u>Transaction and Error Handling:</u>

a. To Maintain Constancy of database we need use transaction so that if any failure will ocuur then either all or no any query reflect in database
   i. For Which Just Have To Write Begin Tran
   ii. Commit tran if No error
   iii. Rollback Tran if any error raise
b. To Throw an we using raiserror function
c. For Error Handling Unlike PL using try and catch (begin try end try, begin catch end catch)
d. To Get Details about error have different function like Error_Numbrer() and many more and for log have store procedure sp_readerrorlog()
e. Also Have Option of @@Error Function But it reset value before every line scan because of which have to maintain seprate variable for that

9. <u>Functions:</u>

a. Three Types of function is there
   i. Scaler
      1) <u>Signature:</u>

```
Create function functionName(para with type)
Returns returntype
```

As
Begin
//code
Return return type variable
End

2) Can Not Return Timestamp, Cursor or Text value

ii. Inline Table Value Function

1) Signature:
Create function functionName(para with type)
Returns table
As
Return (select query)
End
2) Treats as View
3) Changes are affect to base table is change is easy to implement otherwise throws an error

iii. Table Value Function

1) Signature:
Create function functionName(para with type)
Returns @tablename table (paras with datatype)
As
Begin
//code
return
End
2) Treats as a stored Procedure
3) As It is return variable value not as table so Changes are not Possible as Value is assigned in table and returned now to variable is exists so.

10. View

a. View is also called as virtual table.
b. It is very similar to inline table value function, Difference is Can Create Trigger on View but not on function, View can Not Accept Value, Can Create index on View not on table and generally creating index in that case when data is not change frequently like OLAP and if View is indexed then it capable to store the data other then that it is just storing queries and also it is best way to share table with users as it is more secure in compare to direct access of table and whatever changes made is apply to base table is easy to implement otherwise it throw an error like in case of data modification affects to multiple table. Can Create only unique clustered index means Primary Key.

c. Signature
Create view name
As
Select query

d. Can Create, Alter or Drop a view.

11. Stored Procedure

a. It is one of safest way to execute queries.
b. Just a small difference between stored procedure and function is stored procedure can accept input paras and have output para where as functions not have output para as well stored procedure returns only integer not other then that where function can return any type of value.
c. Signature:
Create proc name
Para_list with type [=default value null or 0 based on type] in/out
As
Begin
//code
end
d. Main Advantages of the store proc is One it is complied then plan is set and at runtime or every time no need to compile which saves time and traffic a lot, And Plan reuse every time, Database Logic is separate so debugging of big project is easy.

12. Cursor

a. It is not that much helpful as it read record row by row which is very costly so.
b. It is like pointer which read every row one by one.
c. Signature:

```
declare name  cursor FOR <select Query>
Open name
    fetch Next From name in @var1 @car2
While(@@fetch=0)
    fetch Next From name in @var1 @car2
Close name  // Release the result Set
Deallocate name;
```

13. Dynamic SQL

a. Basically It means Bind Query in such a way that it efficiently work
b. QUOTENAME('value',quote) is used to quote name so that safe from Injection Attack.
c. Signature:
Declare @sql varchar(1000);
Declare @params vrachar(1000);
Append Query with Paraname in @sql (set @sql='select * from @tablename')
Append Para Details in @params (set @params='@tablename varchar(20)')
Exec sp_executeSql @sql,@params,@tableName='tablename'
d. Can Use Exeute() to execute but it only accept sql not other parameter so need to need to bind para values too.

14. More

a. To Get Stored Object Details: select * from sysobjects, select * from Information sys.objects, select * from Information_Schema.
b. To Merge Same Structure:
Merge tablename as t1
Using tablename as t2
On t1.id=t2.id
When matched then
//code
When not matched by target then
//code
When not matched by source then
c. To Clear Plans cache DBCC FREEPROCACHE, sys.exec_cached_plans to get data.
d. SEQUENCE object to Create Counter
i. Signature :
Create Sequence dbo.Name
As datatype number
Start with val number
Increment by number
Minvalue number
MaxValue number
Cycle
Cache number
ii. Next value for dbo.sequence name
iii. To Get Current Sequence current Value from sys.sequences where name=<name of sequence>
iv. To Reset Alter sequence name reeset value

SQL Server

Wednesday, February 10, 2021 10:09 AM

Database

1.	Modify Name	Alter database <old_name> modify name=<new_Name> **or** exec sp_rename oldname,newname
2.	To Get Data Dictionary	In Query Area Select Table Name and press alt + F1 / sp_help
3.	To Allow Identity Column to supply value manually	SET Identity_Insert tablename ON; Note : Have to Supply Values like regular Insertion
4.	Check	Add constraint <nameofc> default(value) for column_name
5.	Default	Add constraint <nameofc> CHECK(expression)
6.	Options For Foreign Key Column	Set Null, Set Default, Cascade.
7.	To Reset Identity	DBCC CHECKIDENT(tblName,RESEED,seed value);
8.	To Get Current Identity Value	• Scope_Indentity() : Of Particular Scope in Same Scope • @@identity () : Same Session but Across Every Scope • Ident_Current('tableName') : OF Particular Table. • Generally All With Select To Display Value
9.	To Get Distinct Value form Column	Select distinct column Name/s from tableName
10.	To Provide Condition (Filter or short)	Select columns from tableName where condition • For Not Equal to <> **or** != • Other Like =, >=, <=, =, >, < • **IN** : where column IN (Lists of Elements) - True for all Value Form List • **NOT :** where column NOT IN (Lists of Elements) - True for all apart from Value Form List **or** satisfy condition like NOT Between and NOT LIKE • **BETWEEN :**where column between value1 and value2 - for Ranging • **LIKE:** where column LIKE 'pattern' ○ % : Like * in REGEX means zero or more occurrence ○ _: Like . In Regex means only one char ○ [] : Like [] in regex any form this ○ [^]: Like [^] in Regex not any from this • **AND & OR :** For Logical • **IS NULL or IS NOT NULL : Compare Null Values**
11.	For Sorting	Select * from column_name order by (column_name ASC /DESC)* • Default ASC
12.	To Retrieve few records of choice	Select top number . . .
13.	To Aggregate	Select . . . . tableName group by columnName/es (Optional: Having=value) • Can Select or Operate Aggregate Functions Only or columnName/es By Which Aggregate. • Using Having We Can Get Particular Values only like Gender is Male • If Condition is there can use Where Having is Helpful to filter based on Value
14.	Types of Join	• Inner Join : Common of A and B • Left Join : Complete A + Null Where Not Matched With B • Right Join : Null Where Not Matched With A+ Complete B • Full Join : Complete A and B (Null Where Not Matched in A or B) • Cross Join : A * B = Each of A is Join With Each of B • Self Join: It is not any different type of join just a concept that if needed then can join same with itself
15.	To Replace Null Value	• ISNULL() function • Case Statement

		• Case when exp then value else value as ColnameToReprsent • COALESCE() function • Generally It Returns First No Null Values from Provided List of values
16.	To Combine the resultant data (Same Structure including data type)	• UNION • Remove Duplication after union data and before displaying • UNION ALL • Gather all data and Display.
17.	Stored Procedure	• With Encryption before definition of stored procedure to stop retrieving definition. • sp_helptext to retrieve definition of stored procedure
18.	Basic String Functions	• **ASCII(char value) :** Gives ASCII Value of character • **Char (int value) :** Character corsponde the ascii value • **LTRIM(string), RTRIM(string), TRIM(string) :** To Trim The Spaces for Left and Right side • **LEN(string) :** To get Length of String • **LEFT/RIGHT(string, int):** To Get Specified Char from string from left/right side • **SubString(string, int, int) :** Substring from string • **Replicate(string, int times) :** To Repeat String times • **Spcae(int) :** To Print Space int times • **Lower(string) :** String To Lower • **Upper (string) :** String To Upper • **Reverse(string):** To Reverse the String • **CharIndex(char, string) :** Returns First Occurrence of char in string, 0 if no • **PatIndex (pattern, string) :** Gives Index where First Occurrence of Pattern is matched, 0 if no • **Replace (string, stringToReplace, ReplacementString) :** To Replace Value • **Stuff (string, startpos, length,replacement string) :** Similar to Replace just way is different
19.	Date Time	• **Data Types:** • Time - 3 to 5 • Date -3 • SmallDateTime - 4 • DateTime - 8 • DateTime2 - 6 to 8 • DateTimeoffset 8 to 10 (Includes Time Zone +/- hh:mm) • Difference is in Accuracy and size of storing 1day/ minute to, 0.0033s - 100 ns • **Functions**: 1. GETDATE / CURRENT TIMESTAMP: Moreover Same 2. SYSDATETIME / SYSDATETIMEOFFSET : Difference is just accuracy 3. GETUTCDATE / SYSUTCDATETIME : To Get UTC based Time 4. ISDATE() - if yes then 1 else 0 (Not Applicable for DateTime2 and offset so max nnn) 5. DAY/ Month/ Year (Valid Date) : Return Particualar Value 6. DateName/ DatePart: (part, date): To Gate More Details About Date in Name or Integer

DatePart	Abbreviation
year	yy, yyyy
quarter	qq, q
month	mm, m
dayofyear	dy, y
day	dd, d
week	wk, ww
weekday	dw
hour	hh
minute	mi, n
second	ss, s
millisecond	ms
microsecond	mcs
nanosecond	ns
TZoffset	tz

7. DateAdd (part,value,date)
8. DateTDiff (part,date1,date2)
9. EOMONTH(date) : End of Month
10. DATEFROMPARTS(year,month,date)
11. DateTIMEFROMPARTS: year, month, day, hour, minutes, s,milis
12. SmallDATETIMEFROMPARTS: year, month, day, hour, minutes
13. DateTIME2FROMPARTS: year, month, day, hour, minutes, s, fractions, precision
14. TIMEfromParts: hh, mm, s, fraction, preciosn
15. DateTIME2(precion)

20.	Cast and Convert	• Both are Use to Convert One Data Type to Another
		  • CAST(column or value as datatype)
		  • Convert(datatype, column or value [style/format])
		  ○ So Just Difference is Can Style Using Convert not By Cast
		  ○ Can Get Styles from MSDN styles for date
		  ○ Cast is More Preferable and Acceptable

21.	Mathematical Functions	• There are Many functions and unlike String Can That from **Programmability/Functions/System Functions/String or Mathematical Functions.** • RAND() : Return float between 0 and 1 and if provide seed always get same value, To Increase Range can increase multiplier. • In ROUND function Third Para Tells Truncate or Round, Minus Places Tells Operation before Point
22.	Functions	• Scaler Function Can Not Return text, Cursor, Timestamp • Inline Tabled Value Function Returns Table and Not Have Begin and End otherwise It Will throw and Error (Returns Table). • In Multi value Table function have to create table variable of Type Table with structure Definition ex: returns @TableName Table (col1 datatype. Col2 datatype.....), and it also need begin and end block, Just Have To write Return • As Returns Table so have to treat like table. • WITH SCHEMABINDING is helpful so that used table can not delectable until Function Is Referring That.
23.	Temporary Table	• Local Temp Table's Scope and Life is till Particular Connection or In Case Of Stored Procedure till execution of Sp only, Can Possible Same Table From Other Connection. • # for Local and ## for Global. • Global Temp Table is till all connection not referring and Unique Across All Users

24.	Indexing	• It is Very helpful as it increase scanning performance, as default table Scanning means row by row every place need to check or handle
		• Sp_HelpIndex tablename : List Indices
		• Types:
		  • Clustered : Only One To Store Data (Default PK)
		  • Non Clustered : As Many but store additional like index page
		  • Unique : To Make Unique Ex: Unique Constraint
		  • Non Unique : if Not Unique Then its is Not-Unique

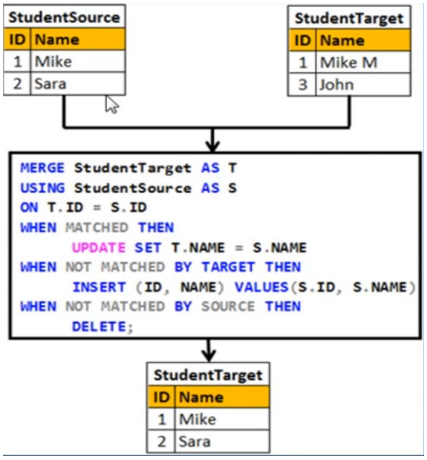
25.	View	• Just a Saved SQL Query or Can Say Virtual Table • As View is not storing Data so we can Update Base Table form View • Can Added Index to view But Main Thing is It is Generally Helpful in case of OLAP as not frequent changes are there in tables so. • Can Not Have Order By Clause • Not Applicable on Temp Tables

26.	Triggers	• Type:
		  • DML : Fires on DML Events like Insert, Update and Delete.
		  ○ After: After Event Occurred

		▪ Maintain Inserted(Inserted or After Update), Deleted(Deleted or Before Update) Table in trigger context only
		○ Instead:
		▪ Specifically to solve operation on view in case of Multiple Tables.
		▪ Update(Column Name) to Check User Updating That Column)
		• DDL
		○ Only For , On Database or On ALL Server
		○ Enable / Disable trigger trDisableCreation on all server
		○ Create_Table, ALter_Table,Drop_Table only
		• Logon
		○ Not A different but Like DDL and Event is LOGON
		`Create trigger tr_AuditLogin`
		`ON ALL SERVER`
		`FOR LOGON`
		`AS`
		`BEGIN`
		`Declare @LoginName nvarchar(100)`
		`Set @LoginName = Original_Login`
		`Select is_user_process, original_login_name, *`
		`from sys.dm_exec_sessions order by login_time desc`
		`END`
		• CLR

27.	Others	• Select * into tablename from tablename, Insert into tblname select * tablename;
		• While(Exists())
		Begin
		End
		• colname from tablename
		• Delete from tblname join tblname2 on : Which Delete All Matched Rows
		• Can Declare Table Variable and use it like table or temp table
		• TOP
		• Object_ID('name') function to check whether Object Exists Or Not
		• To Get All data of Store Proc Can Set Name=@Name or @Name id NULL
		`EXEC sp_settriggerorder`
		`@triggername = 'tr_DatabaseScopeTrigger1',`
		`@order = 'first',`
		`@stmttype = 'CREATE_TABLE',`
		`@namespace = 'DATABASE'`
		`GO`
		•
		`Select is_user_process, original_login_name, *`
		• `from sys.dm_exec_sessions order`
		• Original_Login() to get name of current User
		• Sp_readerrorlog
		• DATALENGTH for Get Size of DataType

28.	Derived Table	• Select id name from () as tblname;
29.	Common Table Expression	• With tblname(column list) as (select) query, tblname(column list) as select query, . ,.... • Is Useful Which immediate Follow by use of that. • Can Change Base Table but if not multiple base then only can change CTE.
30.	PIVOTING	• To Rotate Data one Dimension (Aggregation is Required) • PIVOT (AggregateFunction FOR Column IN Values())
31.	Error Handling	• @@ERROR() • Return 0 if not else Any Error is there • Rollback and commit to commit Transaction

		• Main Thing is it Clear and Reset after Every Statement so May Problematic to Use Directly, need to use local Varibale • Try Cacth • Transaction is in Try Block (Begin Try and End TRY) • Catch Block is After Try Block (Begin Catch and End Catch)
32.	Transaction	• Batch of Commands To Maintain Integrity • Begin Tran or Begin Transaction then Commit Tran or Rollback Tran
33.	Sub Queries	Can Independent or Corelated to Outer Query
34.	Cursor	Declare name cursor FOR select Query Open name Fetch Next From name into @var1 @car2 While(@@fetch_status=0) Fetch Next From name into @var1 @car2 Close name // Release the result Set Deallocate name;
35.	To Get Objects of Database	Object type. Can be one of the following object types: AF = Aggregate function (CLR) C = CHECK constraint D = Default or DEFAULT constraint F = FOREIGN KEY constraint L = Log FN = Scalar function FS = Assembly (CLR) scalar-function FT = Assembly (CLR) table-valued function IF = In-lined table-function IT = Internal table P = Stored procedure PC = Assembly (CLR) stored-procedure PK = PRIMARY KEY constraint (type is K) RF = Replication filter stored procedure S = System table SN = Synonym SQ = Service queue TA = Assembly (CLR) DML trigger TF = Table function TR = SQL DML Trigger TT = Table type U = User table UQ = UNIQUE constraint (type is K) V = View X = Extended stored procedure Select * from SysObjects where XTYPE=' from above list' Select * from sys.objects type Select * from Information_Schema
36.	Re Runnable Query	• Just check Whether Possible or Not before execute Query
37.	Merge	Merge table1 as t1 Using table2 as t2 On Condition Like Join When MATCHED then Code When NOT MATCHED By Target then Code When Not Matched By Source Then Code  All Are Not Compulsory means We Can Choose From That

38.	Except Operator	Just Like Union, It Return Pure Data From A Which Not Contain Shared Data Between Two, It Remove Duplicate But Not In Don't
39.	Intersect Operator	It Return Common Between 2 Tables, Same Duplication
40.	Cross Apply and Outer Apply	Like Inner Join and Left Join, Basically to Join with Table and Table Value Function
41.	More On Select	• Select * Into baktblname from tblname • Baktblname must not exists.
42.	Creation Of User Type	Create type Name as Table (Table Structure of Which we want to create Type)

43.	Passing of table as Para	• Need To Create Type Of That Table
		• Have To Declare Para of That Type With ReadOnly as passed Table Must be ReadOnly No DML Accepts
		`-- Declare the Table Variable`
		`DECLARE @EmployeeTableType EmpTableType`
		
		`-- Insert rows into the Table variable that you want to send to the SP`
		`INSERT INTO @EmployeeTableType VALUES (1, 'Mark', 'Male')`
		`INSERT INTO @EmployeeTableType VALUES (2, 'Mary', 'Female')`
		`INSERT INTO @EmployeeTableType VALUES (3, 'John', 'Male')`
		`INSERT INTO @EmployeeTableType VALUES (4, 'Sara', 'Female')`
		`INSERT INTO @EmployeeTableType VALUES (5, 'Rob', 'Male')`
		
		`-- Pass the table variable as a parameter to the stored procedure`
		`EXECUTE spInsertEmployees @EmployeeTableType`
		• Just Helpful When Want to Use Like Structure Variable Group of Data Types Thats It.
		• To Pass Just have to Create SqlParameter of that Type and Value Is Created Datawith That StructureTable

44.	Grouping Sets	Group By
		Grouping Sets
		(
		  (col1,col2,...),
		  (column List)
		  (May Possible No Columns)
		)
		Also
		  • Order by GROUPING(Column from Group1),GROUPING(Column from Group2)
		  • If We Want Then Can use Rollup or cube as per need To Avoid Manual List of Column
		  • GROUPING Function (column) : if Aggregated then 1 else 0
		  • Grouping ID(List Of Columns) Returns Binary Data Of Individual Grouping

45.	Over Clause	• Aggregate Function(column Name) Over (PARTITION by column name) • By Column Name) • ROW_NUMBER over (order by column name)
46.	Choose Function	• Choose (Index,Values) Like Enum
47.	IIF()	• IIF(exp,Val1,Val2) like Ternary Operator
48.	Sequence Object	• Main Use To Create Automatic Counter • Signature Is Like Create Sequence dbo.Name As datatype number Start with val number Increment by number Minvalue number MaxValue number Cycle Cache number Next value for dbo.sequence name • To Get Current Sequence current Value from sys.sequences where name=<name of sequence> • To Reset Alter sequence name reeset value • Main Thing that it can be shared

49.	GUID	• Unique Identifier (Data Type)
		• NEW ID() to get value
		• 32 char size (16 byte size)
		• Identity is not allowed But Using Default
		• To Create Empty GUID
		`Select CAST(CAST(0 as Binary) as uniqueidentifier)`
		`Select Cast(0x0 as uniqueidentifier)`

50.	Dynamic Sql	`Declare @sql nvarchar(1000)`
		`Declare @params nvarchar(1000)`
		`Set @sql = 'Select * from Employees' + 'Where FirstName=@FirstName and LastName=@LastName'`
		`Set @params = '@FirstName nvarchar(100), @LastName nvarchar(100)'`
		`--Execute sp_executesql @sql, @params, @FirstName='Mark', @LastName='Hastings'`

51.	Plan cache	`SELECT cp.usecounts, cp.cacheobjtype, cp.objtype, st.text, qp.query_plan`
		`FROM sys.dm_exec_cached_plans AS cp`
		`CROSS APPLY sys.dm_exec_sql_text(plan_handle) AS st`
		`CROSS APPLY sys.dm_exec_query_plan(plan_handle) AS qp`
		`ORDER BY cp.usecounts DESC`
		`DBCC FREEPROCCACHE`

52.	Exec() sp_executesql	• EXEC Accept only Ona para • In Sp_executeSQL Can Pass Paras
53.	QuoteName()	• It Put Values in Quote and Table Name in [] to avoid Sql Injection • Quote Name ' <Type Of Quotation Like [or] or ' ' > ' • Undone Effect by PARSE

Note :

- If we directly execute query then all queries got execute but By Selecting and Execute then we can execute selected Query from all list of query in Single Query Window.
- Best way to declare table with tbl prefix and stored procedure with sp prefix not by 'sp_'.
- As to give alias and if Column Name contain Space then [Column Name].
- Main DB Rule is Do Thing As Early as Possible.
- Union Combine the rows of table where as Join combine columns based on logical relationship.
- Inline Table Value Function is batter in performance and sql treat as view, where as Multi Value is treat like Stored Procedure, as Value function can update table which getting from Function.
- Primary Key Uses Clustered and Unique Indexing To Become Unique.
- By Indexing Can Create Constraint.
- As SP, Functions and Views are From Table so signature is like
  - Create or alter proc/function/view name
    Params if (In case of Functions, sp) and Return Type (In Case of function)
    As
    Begin // If Not Inline Table Value Function
    // code
    // return (In Case Of Function)
    End
- In Case of Index and Trigger as are on Table so Signature is Like
  - Create {Type} INDEX / TRIGGER name
    On <tblname or viename> {Colname}
    {These Are for Index Only}
    ########## For Trigger Only #########
    FOR <insert/delete/update>
    As
    Begin
    End
- Raiserror to Throw Error in DB
- Scope of Variable also stored in DB not in memory

- View, Table value function
- Inner Join Two Null as Different Values
- TableName.* To get All From Any Tabel.

Observations:

- In Case of identity when Insert Statement executes then it increment the Identity counter first, so if in case Insert statement is throw an error then also identity counter no get decremented, and next successful will not stored on last stored record's identity + 1 instead of that identity + 1 + in between failed insert Statement. (Not Applicable For Compiling Query)
- If Default is Not Set And Set Default option as Insert Update Specification in FK then it gives NULL.
- As ASCII is till 256 so if provide more then that to char() it return NULL
- Length of Null String is Null same for LEFT and RIGHT, TRIMING functions, Replicate,
- If Date is Not Valid Format then Day, Month and Other will Not work.
- If Multiple Tables are involved in View or inline table value function then Insertion, Updating & deletion may throw error if multiple table get affected.
- String_AGG() function to aggregate strings and in case of pivot if want data which is varchar then min or max is fine.
- May Possible One DDL Trigger affects on Other. Like If We set can to allow create, alter trigger then if we try to create second DDL trigger then not allowing Us to do that.

Confusions and Solutions:

Some **Differences Between VARCHAR** and **TEXT**

The VAR in **VARCHAR** means that you can set the max size to anything **between** 1 and 65,535. **TEXT** fields have a fixed max size of 65,535 characters. ... Meanwhile, **TEXT** is stored off table with the table having a pointer to the location of the actual storage. 19-Feb-2020

The **function** must return a value but in **Stored Procedure** it is optional. Even a **procedure** can return zero or n values. **Functions** can have only input parameters for it whereas **Procedures** can have input or output parameters. **Functions** can be called from **Procedure** whereas **Procedures** cannot be called from a **Function**.
15-Sep-2012

```
SELECT
    c1, c2, aggregate_function(c3)
FROM
    table
GROUP BY ROLLUP (c1, c2);
```

The `ROLLUP` assumes a hierarchy among the input columns. For example, if the input column is (c1,c2), the hierarchy `c1 > c2`. The `ROLLUP` generates all grouping sets that make sense considering this hierarchy. This is why we often use `ROLLUP` to generate the subtotals and the grand total for reporting purposes.

In the syntax above, `ROLLUP(c1,c2)` generates three following grouping sets:

```
(c1,c2)
(c1)
()
```

Useful Links:

- Date and Time Style For Convert : https://www.mssqltips.com/sqlservertip/1145/date-and-time-conversions-using-sql-server/
- Events for DDL triggers in SQL: https://docs.microsoft.com/en-us/sql/relational-databases/triggers/ddl-events?view=sql-server-ver15

Main Thing is Database => Stores Data In Form of Tables and Also Store Other Things Like Stored Procedure, Functions, Views, Indexes, Constraints

Data Storage => Mainly For Data Storing Two Option

1. Temporary Storage (Local Scope Table and Global Scope Table)
2. Permanent Storage (Data Tables)

Tables => Have Columns and Rows Store the Data, Columns May Have Its Own Constraints like
1. Unique (For NULL too, Multiple Nulls are Not Allowed)
2. Check (For Condition Checking)
3. Default value
4. Primary Key
5. Foreign Key
6. Null or Not Null
7. Identity (Auto Increment)

=> Can Add Constraint Inline While Defining Columns
=> Can Add Constraint After Creation of Table too. Using Alter Statement
=> As Table Contains Data So Insert, Update and Delete Statement are There To Handle Data Manipulation
=> Select command is there to display data.
=> Using Alter, Drop, Create can Manipulate Table
=> Using Indexing Can Retrieve smoothly.

Data Types => There are Many Data Types like String, Numbers, Binary, Bit, Date, Time, DateTime
=> To Cast The Data We Also Have CAST(value as dbtype), Convert(datatype, value[style])
=> Predefine Function like For Numeric Round, Floor and Many More, For String Left(), Right() and Many More and For DateTime GETDATE(), YEAR() and Many more

Functions => Can Create Different Functions like Scaler Function, Inline Table Value and Multi-Line Table Value Function
=> Inline Value Function is Treated Like View whereas Multi-Line is Like Stored Procedure.
=> As (INLINE) table Value Function Returning Value So Can Modify Data which is direct replicate to base table, But Main Is Function is Storing Value they just stores query

Stored Procedure => Can Say it is special Kind Of Function Which Have both Input and Output Para, But it is also Storing Queries Which is pre compiled so Not Need To Compile Every time When It Called.

Views => Unlike Function it is also storing Queries but using Triggers we can Change Data Of Base Tables too not matter base tables are more then one and changing affects Multiple Tables (If Designed only then if Not Want to all then Different Case).
=> It is also Called Virtual Table and if data of View not rapidly Got Updated Then Can Create Indexed View Too For Batter Performance. In This case View Is Capable To Storing Data.

Triggers => Triggers are also similar to function and Stored Procedure just main Difference is it execute automatically When Event like Insertion Update and Delete Occurs. Can Not Call Manually.

Aggregation => To Aggregate the data we have group by clause and along with having To Filter Data, Also Aggregate Function Like SUM, Count.

Sorting and Filtering => To Sort The Data Have Order By Clause and For Filter Have Where clause
=> Special Operators like LIKE, IN, NOT, BETWEEN and Other Logical and Conditional Opearators
=> Distinct for Remove Duplication, TOP number to select that much records only.

Joins and Set Operations => Basically Just Small Difference Between Set Operations and Joins is Set Operation Require Same Type of Structure of tables and Joins are Used to Different tables via Foreign Key Primary Key Relation ship to Get More Idea as Data Base is split for normalization
=> Set Operation Include UNION, UNION All, INTERSECT, EXCEPT
=> Different Types of Joins are Inner, Outer (Left, right, Full), Cross Join and Self Join (Basically Not Different Type).
=> Along With This CROSS APPLY and OUTER APLLY is there which are similar to joins but useful in case of Join data of table and data coming from Table Value Function.

Error and Handling of Error => Unlike Programming language we can raise and Handle Error
=> Raiserror() to Raise the error
=> @@ERROR to check error is there or not but before every line scan in get reset so Need To take Casre
=> Try and catch to Handle Error

Derived Table => We Can Derive From table too in case of intermediation storage
=> One Of the Famous way is CTE (Common Expression Table) basically work like view which stores the query but helpful.

Transaction => As Database Must be in Consistent state after every transaction so for consistency we are using transaction so that if any error raise then all transaction get rollback or get execute.

PIVOTING => It is way by which we can rotate Whole Data based on one dimension For Example If Data is Like

Gender	Population

Male	20000
Female	10244

Then PIVOTING ON GENDER

	Male	Female
Population	20000	10244

Handling of Null Values => Can Replace Null Values with no null as part of pre process data using Functions like ISNULL(), COALESCC() and CASE statement.