**70-761**

Create Transact-SQL SELECT queries

1. All the # tables are created in tempdb.
2. While using this statement IF **OBJECT\_ID** ('#AddT','U') IS NOT NULL , use tempdb to check this condition.
3. **U** stands for (**U**ser-defined) table as opposed to system table.
4. **N** stands for u**N**icode (2 bytes) character/string literal.  Same results without the N.
5. Use .. for referring to tempdb, IF OBJECT\_ID **(N'..#AddT'**,N'U') IS NOT NULL.
6. SET RECOVERY bulk\_Logged/Full
7. **ALL** (Compares a scalar value with a single-column set of values.) ALL ( subquery )
8. **EXISTS** ( subquery ) Specifies a subquery to test for the existence of rows.

9. Recovery Models (Alter database ..name… set recovery bulk\_logged/full/Simple with no wait)

Select name,recovery\_models\_desc

From sys.databases;

**SIMPLE** (Can recover to previous full/differential, transactions between that are lost,

No transaction log backup supported

BACKUP LOG TestDatabase to DISK = N’……Path’ (Doesn’t work)

BACKUP DATABASE ……name//// to DISK = /////…Path

Data loss as no logs

**FULL(Default)**

(Has transaction log backups)(First database backup and then log backup)

Take full regularly and log backups regularly

Data loss depends on how regularly log backup taken

**Bulk\_logged** (Keep size of log backup small for some specific minimally logged operations like rebuild clustered index,…… It only marks the data pages that are changed)(Switch to this mode temporarily only)(Overall size is same)

1. Using the **Index Optimizer Hint**
2. **OPTION Clause**

70-761 (kudvenkat)

1. SSMS is client tool. Database server details we connect to server.
2. (local)/./127.0.0.1 for local server
3. Sys dbs (master,tempdb,msdb,model)
4. Create **database/table/view/..object\_Type** name (User defined)
5. Drop **database/table/view/..object\_Type** name (Single user mode)
6. Alter database dbname SET Single\_user with rollback immediate

Two files:

.mdf = Data file (Actual data) (master data file)

.ldf = Transaction log file(Used to recover the database)(log data file)

1. **ALTER Database** db\_name **Modify Name =** new\_name
2. Execute sp\_renameDB ‘Old’,’New’
3. Nvarchar (Unicode,multilingual,2 bytes per char),Varchar(ASCII,1byte per char)
4. Create Table

Create table tblPerson

( ID int primary key,

Name varchar(50) NOT NULL,

Email nvarchar(50) NOT NULL,

GenderID int

)

Create table tblGender

(ID int not null primary key,

Gender nvarchar(50) not null

)

Alter table table\_name add constraint Name\_ss FOREIGN KEY(ID) References tblGender(ID)

1. Default Constraint (Only if you miss value default is taken)

Alter table …..

ADD Constraint……

Default (value) FOR (column)

CHECK (condition)

UNIQUE( column)

**allows nulls but primary key does not** )

(1 primary key but multiple unique keys)

ADD (name) (data type)(null/notnull)

CONSTRAINT (name) DEFAULT (value)

Alter table …..

DROP Constraint……

If null in check,it results unknown which is true/passes

1. In foreignkey,Cascading Referential Intregrity

(No action, Cascade,Set Null, Set default)

CONSTRAINT fk\_name

FOREIGN KEY (child\_col1, child\_col2, ... child\_col\_n)

REFERENCES parent\_table (parent\_col1, parent\_col2, ... parent\_col\_n)

ON DELETE CASCADE

[ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]

1. Identity Column

Id int identity(1,1) primary key (seed,increment)

Identity\_insert on for explicit insert

Else off

Reset seed

Dbcc checkident(‘table\_name’,RESEED,0)

Scope\_identity()(same session/same scope)

@@identity (Same session/across any scope)

Ident\_current(‘Table ame’) (Specific table,any session/scope)

1. SELECT

% - 0/More

\_ = Single

[] - any one

[^] = not one of them

AND/OR operators

Default is ASC(Order by)

Select TOP 10 \* from….

Select top 5 percent \* from ,,,,

1. HAVING (Filter after agg, can use agg,only select)

WHERE (Filter before agg,can’t use agg, select , insert , update)

1. JOINS-----------

Cross Join(Cartesian product of 2 tables)(No ON Clause)

Inner JOIN/JOIN (Remove non matching rows)

Outer (Left,Right,Full)

Left Join/Left Outer Join (Non matching from left)

Right Join/Right Outer Join(Non matching from right)

Full join (Non matching from both tables) (all rows)

Select \*

FROM X INNER JOIN/JOIN Y ON X.a = Y.a

1. Non matching row only

Add where right id is null (left outer)

Where left is null (right outer)

Where any of them is null (full outer)

1. Column IS NULL
2. Self Join-----

Inner Join

Outer (Left,right,full)

Cross Self Join

1. Replace NULL

ISNULL(expression ,New value)

If exp is nULL then new value

CASE WHEN condition THEN …… ELSE ….

END

COALESCE(Column,New Value) returns the first non null value

COALESCE(First,Middle,Last)…….. returns the first non null

Another advantage of COALESCE is that it's a standard function (namely, defined by the ISO/ANSI SQL standards), whereas ISNULL is T-SQL–specific.

With COALESCE, the resulting column is defined as NOT NULL only if all expressions are nonnullable and NULL otherwise. With ISNULL, the resulting column is defined as NOT NULL if any expression is nonnullable and NULL if both are nullable

http://www.itprotoday.com/network-security/patches-performance-what-enterprises-need-know-about-meltdown-and-spectre

21.UNION (Sort distinct , then remove duplicates , slower)

UNION ALL(Faster) (No. of columns, data types,order of columns should be same) **Order by on last select statement**

1. **Stored Procedure** (Don’t use sp\_)

**DROP PROC <name>**

**ALTER PROC >name>**

CREATE PROCEDURE/PROC <nam>

Text Encryption: <WITH ENCRYPTION>

AS

[BEGIN]

[END]

To execute:

Write name of sp

EXEC <name>,@A = ‘’,@B= ‘sadas’

Execute <name>, parameters order imp

DECLARE @result int

EXEC <name>,@A = ‘’,@result output

**WITH PARAMETERS/VARIABLES HAVE @ In the front**

CREATE PROC <name> @A nvarachar(20),

@B int Out/Output

AS  
BEGIN  
END

**VIEW**

Sp\_helptext ‘<name’



*ALTER proc xyz*

*@A nvarchar(50),*

*@B int out*

*AS*

*BEGIN*

*SET @B = 10*

*END*

**DECLARE @result int**

**exec xyz 'Hool',@result = @result out**

**print @result**

**if not out keyword, @result will be null**

1. Sp\_help (view info about sp/any db object)

Sp\_helptext <name> get text

Sp\_depends any db object

1. Return values only int from SP (Integer status variable)

Create proc <name>

As

Begin

**Return (select count(id) from EMp)**

End

*Declare @result int  
exec @result = <name>*

*print @result*

0 success

Non zero failure default

Return can’t be used to return string, only int are returned

Then use output parameters

1. Return vs output

(Only one, multiple)

(Int, any data type)

(Convey success/failure, return other values)

1. SP cache execution plan try to reuse it

Not like ad hoc sql queries which use same execution plan if we run it again

But even slight change space , para 🡪 new execution plan

**Advantages of SP:**

1. **Execution plan retention/reusablity**
2. **Reduce n/w traffic**
3. **Code reusability/maintainability**
4. **Better Security**
5. **Avoid SQL Injection**
6. String functions

ASCII,CHAR,LTRIM,RTRIM,LOWER,UPPER,REVERSE,LEN

LEFT(string,number),RIGHT(string,int),SUBSTRING(string,start,length),

CHARINDEX(search,total\_string,start)

REPLICATE(str,number),Space(no.),Replace(string,oldpattern,new pattern)

Patindex(Patterm,expression) can use wild cards else zero

Stuff(string,start,len,replace str)

1. Date/Time

(100nano)Time hh:mm:ss[.nnnnnnn] 3 to 5

(1 day)Date YYYY-MM-DD 3

(1 min)Smalldatetime YYYY-MM-DD hh:mm:ss

(0.00333 secomd)Datetime YYYY-MM-DD hh:mm:ss[.nnn]

(100nano)Datetime2 YYYY-MM-DD hh:mm:ss[.nnnnnnn]

(100nano)Datetimeoffset YYYY-MM-DD hh:mm:ss[.nnnnnnn] (+/-) hh:mm

GETDATE()(3 nano)/CURRENT\_TIMESTAMP,GETUTCDATE()

SYSDATETIME(),SYSUTCDATETIME() (ALL sys onehave nano7)

SYSDATETIMEOFFSET() (date+time+nano7+01:00)

ISDATE() – datetime2 returns 0, if not valid returns 0, else valid date,time,datetime returns 1

Day/Month/Year(GETDATE()) (01/31/2014)

Datename(day/weekday/month,datetime) returns string

DatePart(Weekday/month/day,date)returns integer ,

**Same as datename**

DateAdd,(day/month/year,20,date)

Datediff(day/month/year,start,end)

1. Updated: **17 July 2006**

Deterministic functions always return the same result any time they are called with a specific set of input values and given the same state of the database. Nondeterministic functions may return different results each time they are called with a specific set of input values even if the database state that they access remains the same.

Agg/string are deterministic except format.

@@,date wale non deter

Rand,cast,convert,isdate (deter nd non deter)

1. Aggegate function(ignore null except count)

Distinct aggregates, for example AVG(DISTINCT column\_name), COUNT(DISTINCT column\_name), MAX(DISTINCT column\_name), MIN(DISTINCT column\_name), and SUM(DISTINCT column\_name), are not supported when you use CUBE or ROLLUP

Avg(all |distinct col) n

Min

Count

Count\_big

Stdev

Stdevp

Var

Varp

Grouping

COUNT(\*) returns the number of items in a group. This includes NULL values and duplicates.

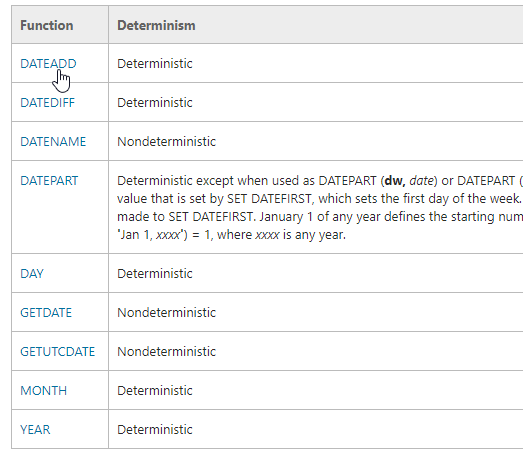
COUNT(ALL *expression*) evaluates *expression* for each row in a group and returns the number of nonnull values.

COUNT(DISTINCT *expression*) evaluates *expression* for each row in a group and returns the number of unique, nonnull values.

Is an aggregate function that causes an additional column to be output with a value of 1 when the row is added by either the CUBE or ROLLUP operator, or 0 when the row is not the result of CUBE or ROLLUP.

Grouping is allowed only in the select list associated with a GROUP BY clause that contains either the CUBE or ROLLUP operator.

|  |  |
| --- | --- |
| **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** |



Precedence:

1. user-defined data types (highest)
2. **sql\_variant**
3. **xml**
4. **datetimeoffset**
5. **datetime2**
6. **datetime**
7. **smalldatetime**
8. **date**
9. **time**
10. **float**
11. **real**
12. **decimal**
13. **money**
14. **smallmoney**
15. **bigint**
16. **int**
17. **smallint**
18. **tinyint**
19. **bit**
20. **ntext**
21. **text**
22. **image**
23. **timestamp**
24. **uniqueidentifier**
25. **nvarchar** (including **nvarchar(max)** )
26. **nchar**
27. **varchar** (including **varchar(max)** )
28. **char**
29. **varbinary** (including **varbinary(max)** )
30. **binary** (lowest)
31. Add (+1 Day, .1 means 2 hours 24 minutes)
32. Create function <name> (@DOB datetime)

Returns nvarchar(50)

As

Begin

End

1. Cast(@ss as nvarchar(2))

Convert(datatype[len] ,exp.[style])

Style number of dates. Ex: 103 dd/mm/yyyy

31. Diff cast vs Convert:

Cast(source as datatype[len][)) ANSI Standard more portability

Convert(datatype([len],source,[style]) Specific to SQL Server,style extra option

103 – dd/mm/yyyy

Style on for nvarchar

32. Abs,Ceiling,Floor,Power,Rand,Square,Sqrt,Round

Rand(seed), fix the pattern🡪 use same seed

Round(number,len,func)

Len = + /- 🡪 (round decimal part/round number before decimal)

Func = 0/1 (rounding/truncation)

Select round(850.566,-1) -> 850.000 , if -2 then 900.000

Select round(850.566,1,1) -> 850.500

33. Functions

**Scalar Functions**(always return a value but not text,ntext,image,cursor,timestamp)

(User defined)

CREATE function <name>(@DOB date,….)

Returns int

As

Begin

Declare @Age int

Return @AGe

End

Invoke but use two part naming/three part naming

Select dbo.<name>(‘ddd’)

**Functions vs proc**

Functions in select,where clause but proc cannot

**Table –Valued**

**Inline table valued(No begin,end)(Like a view , faster than multi statement one)**

**(Parameterized views,can be used anywhere as table,joins and all)**

**(Update the data)**

CREATE function <name>(@DOB date,….)

Returns table

As

Return (select a,v,c from table)

Invoke (select \* from views)

**Multi Statement Table valued(fixed output schema)(Like SP)**

**(Can’t update)**

CREATE function <name>(@DOB date,….)

Returns @T table (Id int,name nvarchar(20))

As

Begin

Insert into @T

Select \* from ….

Return

end

**Deterministic,(same result any time with specifc input value and same state of db) (square,power,sum,avg,count,….)**

**non-deterministic(diff each time called with specific input and db state still same)(getdate,current\_timestamp,….)**

**rand(1) 🡪 deteminsitic**

**rand() -> non-determinstic**

**,with encryption,with schemabinding(odbjects refefrenced by function cannot be changed/deleted)**

**34. Temporary Tables**

Local(#,deleted when connection closed,same name in all possible,rand number added to name,Create SP on it, Temp gets removed once SP is finished, Shared b/w nested SP’s

)

Global(##,when last connection referencing is closed,not possible with same name,rand no not added)

Select name from tempdb..sysobjects

Where name like ‘#personDetails%’

35. Indexes

No index,Table scan

Index on Table8/View

(Index,Row address)

Index Seek is done

Create index <name>

On table name (<columnname> ASC,…c2 DESC..)

**Non unique , non clustered indexes by default**

Sp\_helpindex <tablename>

**Drop index table.index**

**Clustered(**Determines physical order of data in table,only one is possible,primary key automatically creates it unique)

Composite Clustered Index

First drop primary key one, but it does not get deleted using the command

Use object explorer

. Create unique clustered index <name>

On table name (<columnname> ASC,…c2 DESC..)

**Non-clustered(like book index,**data and index at diff places,pointers to data rows,

Asc/descdoesn’t affect data storage,any number)

. Create unique nonclustered index <name>

On table name (<columnname> ,c2)

**Clustered is faster bcoz non-clusters** has to refer back to table to find any column

Clustered don’t require storage whereas non-clustered does

**Unique/Non-Unique**

Add unique constraint,unique index created

PK constraint uses unique clustered index, can’t drop index directly

**ALTER TABLE <anme>**

**ADD CONSTRAINT <ane>**

**UNIQUE CLUSTERED(City)**

Normal UNIQUE (unique,non-clustered)

Index Beneifts in Select,Update,Delete,order by,group by

Where clause also

More indexes,DML can become slow

If all columns of select in index only(Covering Query)

If not, refer table,

Clustered always covers query

Composite index covers a query

**36. Views(Saved SQL query,virtual table,no stored data)(can insert,update,delete)**

**(**Reduce complexity,row/column security,present agg data)

Create view <name>

As

Query

Alter/drop view <name>

Update views based on multiple table(may/may not be correct, INSTEAD OF Triggers are used)

**Indexed Views**

**(First index is always unique clustered , can’t create non-clustered index)**

(Costly,maintain more than table indexs)

(used in OLAP not in OLTP Where data frequently change)

Creating index on views

Make them stored on disk

Like materialized views

Requires:

WITH SCHEMABINDING

Count\_Big(\*) for agg (When group by )

Two part naming

ISNULL to handle nulls

**Limitations of Views:**

**(**Cannot be based on temp tables,

Can’t pass parameters

**Order by is invalid in views,inline functions,derived tables,sub queries,CTE**

unless top,for xml is speicifed

Rules/defauls cannot be used with views)

**37. CTE**

**Want to select count of aggregation >=2?**

**Options:**

1. Direct Having clause
2. Views . then where clause(views stored, but only once it is used)
3. Temp Tables (Global/Local)(Used in nested SP),then where
4. **Table Variables(Passed as parameters b/w proc),**

**Table var are created in tempDB**

**Insert @Var**

**Select ……**

1. Derived Tables(Nested Select statements)(Available in context of current query)
2. **CTE**

**Temporary Result set that is defined within the exceution scope of**

**Single Select,insret,update,delete,create view**

**CTE same as derived table not stored, lasts only for duration of query**

**It is temp result set that can be referenced within a Select,insert,update,delete statement that immediately follows the CTE.**

WITH <name> ([Optional column name list], same no. as in select)

As

(Select statememt (number of col)

),

<name> ([Optional column name list], same no. as in select)

As

(Select statememt (number of col)

)……

Select a,b

From <name>

Where …

**Updatable CTE/VIew**

CTE on 1 base table,update succeeds

CTE on > 1base table, update affects >1 base table, not allowed

**CTE on > 1base table, update affects 1 base table, allowed but not as expected**

**Recursive CTE**

Self Join

CTE References itself is recursive one

With EmployeesCTE(EmployeeID,Name,ManagerId,[Level])

AS

(

**----Amchor**

Select EmployeeID,Name,ManagerId,1

From tblEmplyees

Where ManagerId Is Null

Union ALL

**----- Recursive Member**

Select tblEmplyees.EmplyeeID, tblEmplyees.Name, tblEmplyees.ManagerId, EmployeesCTE.[Level] + 1

From tblEmplyees

JOIN EmployeesCTE

ON tblEmplyees.ManagerId = EmployeesCTE.EmployeeId

) Select EmpCTE.Name,ISNULL(Mgr>name,’Boss’) AS MgrName,EmpCTE.[Level]

FROM EmployeesCTE EmpCTE

Left Join EmployeesCTE MgrCTE

ON EmpCTE.ManegriD = MgrCTE.EmplyeeID

Step 1: Execute the anchor part and get result R0

Step 2: Execute the recursive member using R0 as input and generate result R1

Step 3: Execute the recursive member using R1 as input and generate result R2

Step 4: Recursion goes on until the recursive member output becomes NULL

Step 5: Finally apply UNION ALL on all the results to produce the final output

38. **PIVOT**

Convert unique values in a col into multiple columns of output.

Select unpivt col,[First pivot],[second pivot],…….

FROM

(Select cols to be used produces that data(**3 things)**

From table)

As Source

PIVOT(

(Aggregation)(col being agg)

FOR column that conatins the values to become col headers

IN (First pivot,second pivot,,,,,,)

)AS pivot\_table Order by

Select salesagent,india,us,uk

From prod

PIVOT

(

Sum(amount) for Salescountry in ([india],[us],[uk[)

)as ptable

**Three things:**

1. Column whose value is be aggregated or calculated (SalesAmount)
2. Column across whom value will be agg(SalesAgent)(Unpivot Column)
3. Column whose values will become new columns (SalesCountry)

**40. Error handling**

**@@Error (System function)**

(Using if non-zero then error else not,set to zero at each statement)

If(…)

Begin

**Raiserror(‘Message’,Severity,State)(Sev is 16,state is from 1 to 255)(only geberates from 1to 127)**

End

16 for general errors that can be corrected by user

**Transaction**

*Begin tran*

*If….error*

*begin*

*Rollback transaction*

*end*

*Else*

*begin*

*Commit tran*

*End*

***41. TRY… Catch…(cannot be used in User defined functions)***

Begin try

Begin transaction

(Set of statements that can have error)

Commit transsation

End try

Begin catch

(any other set of statements)(rollback)

**Select (Only work in catch outside they are NULL)**

**Error\_number**

**Error\_message**

**Error\_Procedure**

**Error\_state**

**Error\_severity**

**Error\_Line**

End catch

(any other set of statements)

**42.Transactions**

Group of statements , single unit, either all or none,ACID,maintain integrity

Atomic,Consistent,Isloated,Durable

Read uncommitted data

**SET TRANSACTION ISOLATION READ UNCOMMITTED**

**By default it is committed level**

1. **Atomic (All or none)**
2. **Consistent(Left in logically consistent state)**

**I-Isloation(Locking mechanism,begin and lock that row from other user)**

**D-Durable(change once done is permanent, if power failute, it should rollback)**

**43. Subqueries**

Query inside another query

Returns a signle value and can be nestrd inside a select,update,isnert,delte. (Upto 32 levels nesting). Inner/Outer Queryr, **columns in inner cannot be used in outer. Reverse possible**

**43. Correlated Subqueries**

**Inner** querry depends on outer one, for each row of outer inner is excecuted,

**Subquery in select clause**

Select Name,(Seelct sum(sss) from sales where id =T.id)as Total

From T

Order by name

Existence Check of tables

**If (exists(select \* from information\_Schema.tables**

**Where table\_name = ‘tblEmplyee’))**

**Begin**

**Drop table tblemlpoyee**

**End**

**Performance Joins vs Subquery**

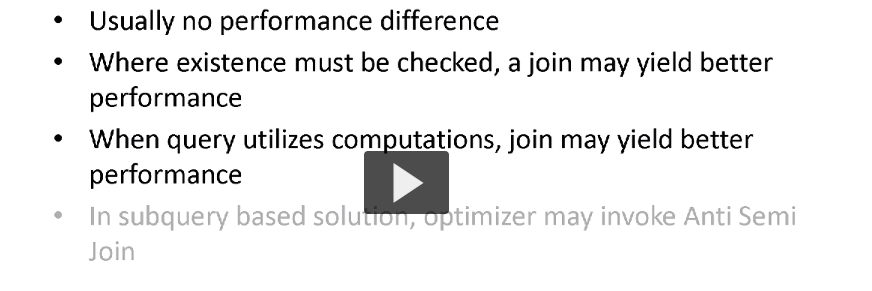
According to MSDN, in sql server, in most cases, there is usually no performance difference

In **general joins work faster than sub-queries**, but in reality it all depends on the execution plan that is generated by SQL Server. It does not matter how we have written the query, SQL Server will always transform it on an execution plan. If sql server generates the same plan from both queries, we will get the same result.

**Most cases –** No performance diff

**In cases when existence is checked -**  **Join better**

In **general joins work faster than sub-queries**, but in reality it all depends on the execution plan that is generated by SQL Server.



**44. Cross Apply and Outer Appply**

**Apply** used to join b/w table and table valued function.

**For** each row value of table(outer), function on right is computed

**No**  need of join condition

**Cross Apply**  is same as inner join

**Outer Apply** is same as Left Outer Join,umat5ched are NULL.

**Select T.DeptNmae,E.Name.E.Gender.E.Salary**

From DeptTable T

Cross Apply dbo.EmployeeTable(Depttable.id) E

Select \* into Table from… (only newtable)

Insert into table , select \* from Employees

**Having (after agg filter groups can be used for where also),Where (before agg filter rows), Having is slower**

**99,100**

**46. Grouping Sets (From 2008)**

**By country,gender**

Select country,gender,sum(salary) as Total

From Employess

Group by Country,Gender

UNION ALL

**Only Country**

Select country,NULL,sum(salary) as Total

From Employess

Group by Country

**Only Gender**

**UNION ALL**

Select NULL,Gender,sum(salary) as Total

From Employess

Group by Gender

UNION ALL

**Grand Total**

Select NULL,NULL,sum(salary) as Total

From Employess

**Using Grouping Sets**

Select country,gender,sum(salary) as Total

From Employess

Group by

**GROUPING SETS**

**(**

**(Country,Gender), (By both)**

**(Country), (by country)**

**(Gender), (By gender)**

**() (Grand Total)**

**)**

Order By Grouping(Country),Grouping(Gender),Gender

**47. ROLLUP (n+1)**

Aggregate op on multiple levels in a heirarchy

(Country,Gender) (2+1 = 3)

**(Country)**

**()**

Grand total+ Normal Group By + (Heirarchy values)

(Country,Gender,cIty) (3+1 = 4)

**(Country,gender)**

**(counry)**

**()**

**ROLLUP(sdasd.asdasd)**

**GROUP BY asdas,sadas WITH ROLLUP**

**48. CUBE(2^n) (All combiantions)**

(Country,Gender) (2^2 = 4)

**(Country)**

**(gender)**

**()**

**CUBE(sdasd.asdasd)**

**GROUP BY asdas,sadas WITH CUBE**

**49. Grouping function (Gives 1 when column in a group by list is agg else 0)**

CASE When grouping(col) = 1 THEN ‘All’ ELSE ISNULL(Counrtry,’Unknown’) END

Can directly use ISNULL but won’t work when NULL values in source table

**50. Grouping ID function (get level of grouping)(Same col list as in GROUP BY to be used in Grouping ID also for sure.)**

Grouping ID (List A,B,c (**Group by col list**) = Grouping(A) + Grouping(B)+ GHroupin(C) = (Decimal Result)

Put a condition on Grouping ID for getting the exact level of agg data.

**51. OVER Clause**

We cannot select unagg col with agg col, it gives an error

To do that, we have to use OVER clasue

**Function() OVER (PARTITION By Col)**

**Break data into partitions.Function executed for every partition**

**Function can be Count.,avg,sum,.min,max,row\_number(),rank,.dense\_rank…..**

**Exceutiion order**

FROM

ON

JOIN

WHERE

GROUP BY

WITH CUBE or WITH ROLLUP

HAVING

SELECT

DISTINCT

ORDER BY

TOP

52. **ROW\_NUMBER() OVER ([Partition BY Gender] Order BY col1,col2)**

Can also use partiution by here

(Order by clause is mandatory)

**Remove Duplicates using this,**

**Partition and order by id**, then delete all those where row\_number > 1

**53. Rank and Dense\_rank function**

Order by is req

Partition in optional

Dense\_rank does not skip ranking

Rank skips if equal found

RANK() OVER (Order by Col1,CLol2,. ASC/DESC.) 1,1,3,4,5

DENSE\_RANK() OVER (Order by Col1,CLol2,..) 1,1,2,3,4

Each partition sets rank to 1 again.

NTh highest salary.

Better to use dense\_rank

If (n-1)th was same, rank will skip the nth and we won’t find it.

With result as

(

Select salry,dense\_rank() over (order by salary desc) as salary\_rank

From employees

) select TOP 1 Salary from Result wehe salary\_rank = 2

**54. Running Total**

Select Naem,

SUM(salary) OVER ([Partiton By asdas ]Order By ID(**Use unique thing in order by**))AS RunningTotal

FROm T

55. **NTILE(Number) OVER (Order by col1,col2.,,,,)**

Order By (Req), partition by is optional,distributes into specified no. of groups

Not div then unequal but larger comes first

No = 2 , 10 rows into 5 each

No= 3 ,10 == 4,3,3

No = 11 , 10= 1,2,3,4,5,6,7,8,9,10

**Lead(Col,[offset],[Default]) OVER (Order By col1,,,,) (Subsquent row data along with current)**

**Lag(Col,[offset],[Default]) OVER (Order By col1,,,,) (Previous row data along with current)**

Default offset =1 , default is NULL

Partition is optional

**First\_Value(col) OVER (Order by Col1…..)(get first value from col)**

**Gives first value in column (very simple)**

**Last\_Value(col) OVER (Order by Col1…..)(get first value from col)**

**56. Window Functions in SQL Server**

**Agg –** Sum,avg,count,…

**Rank -** Rank,Dense\_rank,row\_number

**Analytic -** Lead,Lag,first\_value,last\_value

**PARTITION BY**

**Order BY**

**Rows/Range**

**Default is Range/Rows between unbounded preceding and current row**

Rows between unbounded preceding (first row) and unbounded following(last row)

Rows between 1 preceding(1 preceding) and 1 following (1 next row)

Rows vs Range

**ROWS/Range same until duplicate rows.**

**Rows (duplicates are distinct values)**

**Range(they are single enitity)**

**Range Rows**

**1000** 2000 1000

**1000 2000 2000**

**3000 8000 5000**

**3000 8000 8000**

**5000 13000 13000**

**57. UNPIVOT (Columns into rows)**

Select SalesAgent,Country,SalesAmount

FROM Source

UNPIVOT

(

SalesAmount

FOR Country IN (india,US,UK)

) AS Unpivot\_Table

**Reverse PIVOT (Always possible?? Only if pivot has not aggegated**

**the data else no)**

**ALTER TABLE**

[**https://www.youtube.com/watch?v=t2baJaomY4o**](https://www.youtube.com/watch?v=t2baJaomY4o)

**58.Temporal tables(2016) (System Versioned tables) (6th SCD)**

Way to store the history data for auditing and analyzing purposes

Mirror schema of history table(No contrsintsn,PK,FK,table constraings)

Can’t drop/truncate until versioning = OFF

Insert/update/delete in table, data moved to history table with time range.

INSTEAD OF not allowed

INSERT,UPDATE can’t use System\_time

Things mandatory:

1. Primary Key
2. Period Columns (ValidFrom, ValidTo)

Use cases:

1. Create a new temporal table

Create table dbo.Emp

(

ID int identity(1,1) primary key clustered,

Data nvarchar(50) not null,

[Start] datetime2 generated always as row start,

[End] datetime2 generated always as row end,

period for system\_time([Start],[end])

)

with (system\_versioning = ON (HISTORY\_TABLE=dbo.EmpH ))

1. On an old table

Create table dbo.EmpT

(

ID int,

Data nvarchar(50) not null

)

Alter table dbo.EmpT

ALTER COlumn ID int NOT NULL

Alter table dbo.EmpT

ADD CONSTRAINT ID\_PK PRIMARY KEY(ID)

Alter table dbo.EmpT

ADD

[Start] datetime2 generated always as row start default CAST('1900-01-01' AS datetime2) ,

[End] datetime2 generated always as row end default CAST('9999-12-31' AS datetime2),

period for system\_time([Start],[End])

Alter table dbo.EmpT

SET (system\_versioning = ON (HISTORY\_TABLE=dbo.EmpTH ))

Direct can’t change history data

Put version = off and then drop both the tables.

How to Query Temporal Tables??( <https://www.youtube.com/watch?v=JWPgc8zHMuM>)

Select \* from table for system\_time …….. where ……

**AS OF**<date\_time> (SysStartTime <= date\_time AND SysEndTime > date\_time)

(Includes all rows that were active at that moement in time,includes that become active at that moment but excludes that ceased to be active at that moment)

**FROM**<start\_date\_time>**TO**<end\_date\_time>   
SysStartTime < end\_date\_time AND SysEndTime > start\_date\_time)

**(Don’t matter when it was active or when ceased it was active during this interval)**

**BETWEEN**<start\_date\_time>**AND**<end\_date\_time> (SysStartTime <= end\_date\_time AND SysEndTime > start\_date\_time)

(Includes thoe that started at end data time)

**CONTAINED IN** (<start\_date\_time> , <end\_date\_time>) (SysStartTime >= start\_date\_time AND SysEndTime <= end\_date\_time)

(Active and closed between this range only)

**ALL (All rows)**

**(all rows)**

query and output JSON data, query and output XML data

**JSON**

**Use**

**JSON\_VALUE**(Extract the value from JSON text) (Selcect,where,group by)

**JSON\_VALUE(@json,’$.Order.price’) 🡪 NVARCHAR(4000)**

**JSON\_QUERY,**(Extract JSON fragment from JSON text) (Objects/Arrays)(Get the JSOn object)

**(Can’t return value, it will say NULL)**

**JSON\_QUERY(@json,’$.Order.Items[2]’) 🡪 NVARCHAR(MAX) if not specify anything path, it returns original object**

**JSON\_MODIFY(Update** delete,add proerties in JSON text)

(Makes a copy not inplace)

**(returns the new JSON with updated values)**

JSON\_MODIFY(@json,’$.Order[5].Price’,1) 🡪 NVARCHAR(MAX)

If that field not there, it will insert.

Can append values to the array

JSON\_MODIFY(@json,’append $.Teams’,”Shiwal’) 🡪 NVARCHAR(MAX)

For delete pass NULL values

JSON\_MODIFY(@json,’$.Order[5].Price’,NULL)

Insert JSON into this….

JSON\_MODIFY(@json,’$.Born’,JSON\_QUERY(‘{sddd…….})

It know josn\_query wll return valid json. Otherwise it will input plain text and not json.

**ISJSON** ---- (Check valid JSON or not the text is formatted as JSON or not)

(ISJSON(N’{“min-price”:120,”Max-price:220}’) -> 1/0

OPENJSON to table

Table to JSON use FOR JSON

**OPEN JSON**(Get the Key value pairs from Object/Array )table valued function that takes JSON array/collection, define a schema of the table and compute the json object to array) (It directly returns just the key value pairs with type unless we specify the WITH clause and JSON PATH givern)

**Array gives key index too**

Array of JSON objects to table

Every object to one row.

Every Key- value pair to corresponding cell and value in a row.

Select \*

FROM OPENJSON(@json,N’$.Orders’)

WITH (

Number varchar(200) [N’$.Order.Number’] ----------- **Optional, only use it for nested properties**

Date datetime N’$.Order.Date’, (can rename the col once you specify the path otherwise key name)

Customer vartchar(200) N’$.Account,

Quanity int N’$.Item.Qty’,

)

***Get the 4000 limit words:***

Select value from OPENJSON(@json) where [key]= ‘Bio’

Select Bio from OPENJSON(@json) WITH (Bio nvarchar(MAX))

**Data type can be json also**

***With***

***(period nvarchar(max) as JSON***

)

Select \*from

OPENJSON(@json,’Sborn;)

WITH (

ABC nvarchar(200),

Bio nvarchar(MAX))

**FOR JSON (**Table to JSon array)(Define col names as the hierarchy you require and make them nested using dot structure)(Result of select query as JSON)

**Select**  Number as [Order.Number], Date as [Order.Date],

**Customer as Account**

**Price** as [Item.Price]

**FOR JSON PATH*,ROOT(‘Orders’) (One additional nesting level)***

**Select 1 as x,2 as y,3 as z, null as nothing**

For json path,[ without\_array\_wrapper]

[{‘x’:1,’y’:2,’z’:3}]

**Remove aaray use without\_array\_wrapper (If multiple records you** without\_array\_wrapper, then take the result and concat with other result bcoz its not valid JSON) (Gives invalid JSON,use JSON\_Query)

**{“x”:1,”y”:2},{“x”:3,”y”:4}**

**{“object”:[{“x”:1,”y”:2},{“x”:3,”y”:4}]} (adding one more level )**

**Select 1 as [point.x],2 as [point.y],3 as z, null as nothing**

For json path

With src(x,y) as (select 1 as x,2 as y union select 3 as x, 4 as y)

Select \* from src for json path;

Like JSON in select statement

**Select 1 as [point.x],2 as [point.y],3 as z, JSON\_QUERY({“x”:1,”y”:2}) as json**

For json path

Add to the array

“Json”:{x:2,y”3}

Nested json using nested select statements inplace of JSNO\_QUEY above write selce statemnetn.

**Strict mode**

Slect JSON\_VALUE(@json,**’strict $.Item.Price’)**

If not found , it says nULl but use strict to get the error

(Proprty cann’t be found in specific PATH)

Select JSON\_VALUE(@json,**’strict $.Bio’)**

It is an object and not a single value that’s why it said so, it says nULl but use strict to get the error

(Scalar value cannot be found in specified PATH)

Select JSON\_QUERY(@json,**’strict $.ITEm,price’)**

It is an single value and not object/array that’s why it said so, it says nULl but use strict to get the error

(Object/Array cannot be found in specified PATH)

**Things to keep in mind in JSON:**

**(Is plain text, Case sensitive)(Array index starts from 0)**

**(If space in key name , use double quotes)**

**(If > 4000 in JSON\_VALUE it says NULL)**

**(If use JSON\_Value to get the object, and not a values/property it will give NULL)**

**XML**

----- Data Types (XML data type that stores data within a column)

AdditinalCntat (XML(Person,additioncontactingoSchemaCollection),null)

**Add schema definition sheets what tags can be used within XML column.**

----- Retreiving XML from SQL

Importing XML Into SQL

OpenXML (Display XMl into rowset)

ForXML (Display results into XML)

Select first,seconm

FRom P

FOR XML AUTO,ROOT(‘People’),Elements

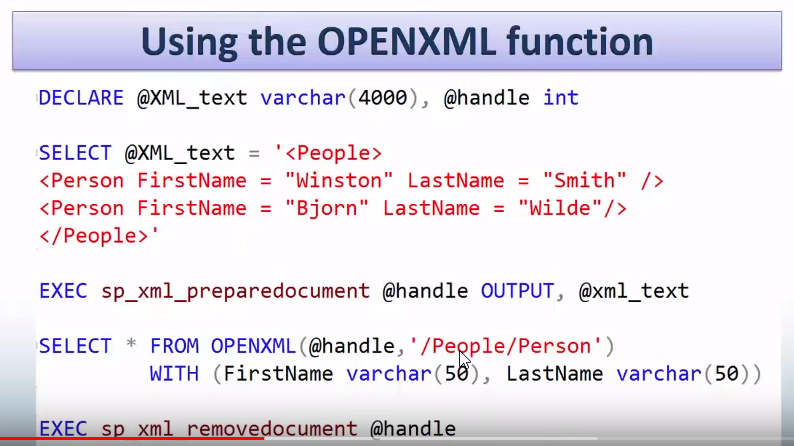
Four options:

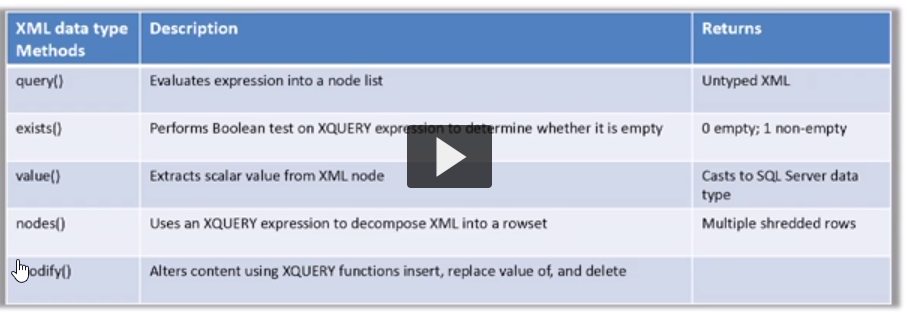
FOR XML RAW

FOR XML AUTO

FOR XML EXPLICIT

FOR XML PATH

roc

****

@handle to store the int valur of current location in memory where my data is located.

Select name

from sys.tables aS SP

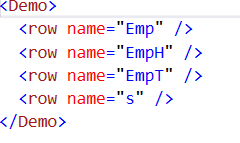
for xml raw,root('Demo') ,[elements]

RAW With elemnts same as PATH but tags have row

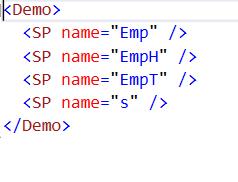
AUTO with elemtns same as PATH but with SP in tags

PATH with elements same

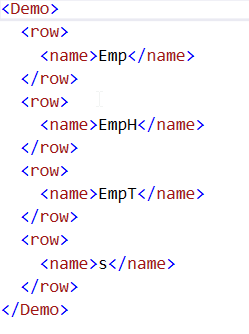
RAW

****

**AUTO**

****

PATH



XML

Things to learn:

<https://docs.microsoft.com/en-us/sql/t-sql/statements/insert-transact-sql>

**1,2,3,4,5,6**

**15**

**19**

**41, 42**

**46,47,48, 49. out of these 4 (46-49) only 3 came. but i didnt remember which 3.**

**54 to 60**

**Dumps**

1. SET XACT\_ABORT On – whole transaction is terminated and rolled back.

If OFF - maybe whole sometimes but sometimes only statement is rolled back

(when runtime error is raised)(set at runtime)(not compile errors)

1. XACT\_STATE (0(no active request from user),

1 (Active request and is capable tof commmiting and wirting)

-1 (uncommittable tran,full rollback needed, no read werite, error message send to client)

3. @TRANCOUNT (Begin +1 ,Commit -1 , Rollback reset to 0)

4. BEGIN

(transaction\_name| @tran variable)(outermost pair use it only)(32 limit)

(with mark [‘description’])( WITH MARK option causes the transaction name to be placed in the transaction log)( WITH MARK is used, a transaction name must be specified)

(Outermost begin tran with mark only applies)

5. Commit (tran name WITH ( DELAYED\_DURABILITY = { OFF | ON })

\*Only when outermost is committed , all other are committed really(When tran count is 0)

6. Rollback (tran name or savepoint\_name)(to begin /savepoint inside transaction)

(Rollback nesting, outer one rollbacks , inside even if committed are rolled back)

1. RaiseError(‘Message’,sev(0-18,19-25),state(0-255))

RAISERROR ( { msg\_id | msg\_str | @local\_variable }

{ ,severity ,state }

[ ,argument [ ,...n ] ] )

[ WITH option [ ,...n ] ]

(11 to 19 push to catch block)

(When msg\_id is not specified, RAISERROR raises an error message with an error number of 50000.)

Is an integer from 0 through 255. Negative values default to 1. Values larger than 255 should not be used

Generates an error message and initiates error processing for the session

Shoule be valud msg\_id in sys.messages

**For severity levels from 19 through 25, the WITH LOG option is required.**

|  |  |
| --- | --- |
| LOG | Logs the error in the error log and the application log and MS SQL Alert is raised. for the instance of the Microsoft SQL Server Database Engine. Errors logged in the error log are currently limited to a maximum of 440 bytes. Only a member of the **sysadmin** fixed server role or a user with ALTER TRACE permissions can specify WITH LOG. |
| NOWAIT | Sends messages immediately to the client. |
| SETERROR | Sets the @@ERROR and ERROR\_NUMBER values to *msg\_id* or 50000, regardless of the severity level. |

1. THROW number,message,state(tinyint)

(Raises an exception and transfers execution to a CATCH block )

Any number > 50000

Message(2048 no printf)

State (0 to 255)

No severity like raiserror (default is 16)

If the THROW statement is specified without parameters, it must appear inside a CATCH block. This causes the caught exception to be raised. Any error that occurs in a THROW statement causes the statement batch to be ended.

BEGIN TRY

  DECLARE @RESULT INT = 55/0

END TRY

BEGIN CATCH

  PRINT 'BEFORE THROW';

  THROW;

  PRINT 'AFTER THROW'

END CATCH

  PRINT 'AFTER CATCH'

THROW returns correct error number and line number and also the PRINT statement before the THROW statement must be followed by the semicolon (;).

1. INSERT INTO single sattament two values to ensure both or none.
2. (INDEXED VIEW)WITH SCHEMABINDIN

You can not use *SELECT \** in a schemabound view.

1. STRING\_SPILT(str,separator) returns col of values (table valued function) (The name of the column is **value**. )
2. RAISEERROR VS THROW

(Creates error message) , (raised exception move to catch)

(It can use sys definded excpetions) , (only above 50000)

(It cannot get original message) , (It can get original message)

(Print after) , (print before)

1. The TRY…CATCH construct cannot be used in a user-defined function.

TRY

(doesn’t send 10 or lower ones to catch,20 or higher but handles if connection doesn’t break)

CATCH(

Does not handle syntax errors, name resoluition errors)

If an error occurs during compilation or statement-level recompilation at a lower execution level (for example, when executing sp\_executesql or a user-defined stored procedure) inside the TRY block, the error occurs at a lower level than the TRY…CATCH construct and will be handled by the associated CATCH block.

1. <https://docs.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql>
2. Ansswer Dump25 :

**DELETE Sales.Orders FROM Sales.Orders WHERE OrderDate < '20120101' AND ShippedDate IS NOT NULL**

(Doubt20) ---- Doubt

1. Dump27

*SELECT Complaints.ComlaintID, Persons.Name  
FROM Persons  
JOIN Contacts  
ON Persons.PersonID=Contacts.PersonID  
JOIN Complaints  
ON Contacts.ComplaintID=Complaints.ComplaintID*

1. Dump26

*SELECT TOP(3) FirstName + ‘ ‘ + LastName AS Fullname, S.SalesYTD  
FROM Person as P INNER JOIN SalesPerson AS S  
ON P.PersonID = S.SalesPersonID  
WHERE S.TerritoryID IS NOT NULL  
ORDER BY S.SalesYTD*

1. OUTPUT INTO Clause (Returns information from,expressions based on, each row affected by INSERT,UPDATE,DELETE,MERGE)

INSERT INTO Table

OUTPUT Inserted.\*

INTO @Table

VALUES (…..)

DELETE

OUTPUT Deleted.\* INTO

FROM

WHERE

UPDATE

SET

OUTPUT INSerted (New),Delte(Old)

INTo

FROM

WHERE

1. IIF(Condtion.True,False)
2. Besides, null values in the input of UNPIVOT disappear in the output, whereas there may have been original null values in the input before the PIVOT operation.
3. Another advantage of COALESCE is that it's a standard function (namely, defined by the ISO/ANSI SQL standards), whereas ISNULL is T-SQL–specific.
4. Doubt36
5. A TRY\_CONVERT function returns a value cast to the specified data type if the cast succeeds; otherwise, returns null. References: <https://msdn.microsoft.com/en-us/library/hh230993.aspx>

Same as CONVERT(datatype.,value) but handles case when failed

1. *SELECT CityID, QuestionID, RawCount  
   FROM Cities AS t1  
   (SELECT Tokyo, Boston, London, “New York” FROM Rawsurvey) p  
   UNPIVOT  
   (Rawcount FOR CityName IN (‘Tokyo’,’Boston’,’London’,’New York’)  
   AS t2  
   JOIN t2  
   ON t1.CityName = t2.cityName*

Doubt43

1. Except(Minus A - B) and Interest(Intersection)
2. , if a WHERE clause is added, the cross join behaves as an inner join
3. EXCEPT returns distinct rows from the left input query that aren’t output by the right input query.
4. INTERSECT returns distinct rows that are output by both the left and right input queries operator.(NULL Values problem)
5. References: <https://msdn.microsoft.com/en-us/library/ms178107.aspx> (XML)
6. JSON (Read thought it)
7. NEWID() globally unique random value

NEWSEQuentialid() sequential unique

1. DATEFROMPARTS
2. Temporal

SYSTEM\_VERSIONING = ON , MEMORY\_OPTIMIZED = ON, DURABILITY = SCHEMA\_AND\_DATA,

(HISTORY\_TABLE = dbo.Department\_History, DATA\_CONSISTENCY\_CHECK = ON,

34. ALTER TABLE dbo.doc\_exz

ADD CONSTRAINT col\_b\_def

DEFAULT 50 FOR column\_b ;

ALTER TABLE dbo.cnst\_example NOCHECK CONSTRAINT salary\_cap;(Disable) CHECK for enable.

35.cte

The following clauses cannot be used in the *CTE\_query\_definition*: 

* COMPUTE or COMPUTE BY
* ORDER BY (except when a TOP clause is specified)
* INTO
* OPTION clause with query hints
* FOR XML
* FOR BROWSE

Anchor members must be combined by one of these set operators: UNION ALL, UNION, INTERSECT, or EXCEPT. UNION ALL is the only set operator allowed between the last anchor member and first recursive member, and when combining multiple recursive members.

The following items are not allowed in the *CTE\_query\_definition* of a recursive member: 

* SELECT DISTINCT
* GROUP BY
* HAVING
* Scalar aggregation
* TOP
* LEFT, RIGHT, OUTER JOIN (INNER JOIN is allowed)
* Subqueries
* A hint applied to a recursive reference to a CTE inside a *CTE\_query\_definition*.

36. View not alllowed

COMPUTE or COMPUTE BY clauses

An ORDER BY clause, unless there is also a TOP clause in the select list of the SELECT statement

The INTO keyword

The OPTION clause

A reference to a temporary table or a table variable.

37. Null values are treated as the lowest possible value (Order by)

38. gROUPING SETS perform UNION ALL actually

39. Distinct aggregates. The aggregates AVG (DISTINCT column\_name), COUNT (DISTINCT column\_name), and SUM (DISTINCT column\_name) are supported with ROLLUP, CUBE, and GROUPING SETS

### 40. **Querying results of a FOR XML query**

**41. FOR SYSTEM\_TIME** can be specified independently for each table in a query. It can be used inside common table expressions, table-valued functions and stored procedures.

42. OFFSET 5 rows ofetch 10 rows only

43. cast,convert,parse, and tryforall three.

44. 