# 3. Table Structure

select \* from tblDepartment

select \* from tblEmployee

select \* from tblTransaction

select min(EmployeeNumber) as MinNumber, max(EmployeeNumber) as MaxNumber

from tblTransaction

select min(EmployeeNumber) as MinNumber, max(EmployeeNumber) as MaxNumber

from tblEmployee

# 4. Subquery – WHERE

select T.\*

from tblTransaction as T

inner join tblEmployee as E

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeLastName like 'y%'

order by T.EmployeeNumber

select \*

from tblTransaction as T

Where EmployeeNumber in

(Select EmployeeNumber from tblEmployee where EmployeeLastName like 'y%')

order by EmployeeNumber

# 5. Subquery – WHERE and NOT

select \*

from tblTransaction as T

Where EmployeeNumber in

(Select EmployeeNumber from tblEmployee where EmployeeLastName not like 'y%')

order by EmployeeNumber -- must be in tblEmployee AND tblTransaction, and not 126-129

-- INNER JOIN

select \*

from tblTransaction as T

Where EmployeeNumber not in

(Select EmployeeNumber from tblEmployee where EmployeeLastName like 'y%')

order by EmployeeNumber -- must be in tblTransaction, and not 126-129

-- LEFT JOIN

# 6. Subquery – WHERE and ANY, SOME and ALL

select \*

from tblTransaction as T

Where EmployeeNumber = some -- or "some"

(Select EmployeeNumber from tblEmployee where EmployeeLastName like 'y%')

order by EmployeeNumber

select \*

from tblTransaction as T

Where EmployeeNumber <> any -- does not work properly

(Select EmployeeNumber from tblEmployee where EmployeeLastName like 'y%')

order by EmployeeNumber

select \*

from tblTransaction as T

Where EmployeeNumber <> all

(Select EmployeeNumber from tblEmployee where EmployeeLastName like 'y%')

order by EmployeeNumber

select \*

from tblTransaction as T

Where EmployeeNumber <= all

(Select EmployeeNumber from tblEmployee where EmployeeLastName like 'y%')

order by EmployeeNumber

-- anything up to 126 AND

-- anything up to 127 AND

-- anything up to 128 AND

-- anything up to 129

-- ANY = anything up to 129

-- ALL = anything up to 126

-- any/some = OR

-- all = AND

-- 126 <> all(126,127,128,129)

-- 126<>126 AND 126<>127 AND 126<>128 AND 126<>129

-- FALSE AND TRUE = FALSE

-- 126 <> any(126,127,128,129)

-- 126<>126 OR 126<>127 OR 126<>128 OR 126<>129

-- FALSE OR TRUE = TRUE

# 7. Subqueries in the FROM clause

select \*

from tblTransaction as T

left join (select \* from tblEmployee

where EmployeeLastName like 'y%') as E

on E.EmployeeNumber = T.EmployeeNumber

order by T.EmployeeNumber

select \*

from tblTransaction as T

left join tblEmployee as E

on E.EmployeeNumber = T.EmployeeNumber

Where E.EmployeeLastName like 'y%'

order by T.EmployeeNumber

select \*

from tblTransaction as T

left join tblEmployee as E

on E.EmployeeNumber = T.EmployeeNumber

and E.EmployeeLastName like 'y%'

order by T.EmployeeNumber

# 8. Subquery – Select Clause

Select \*, (select count(EmployeeNumber)

from tblTransaction as T

where T.EmployeeNumber = E.EmployeeNumber) as NumTransactions,

(Select sum(Amount)

from tblTransaction as T

where T.EmployeeNumber = E.EmployeeNumber) as TotalAmount

from tblEmployee as E

Where E.EmployeeLastName like 'y%' --correlated subquery

# Remainder

select \*

from tblTransaction as T

Where exists

(Select EmployeeNumber from tblEmployee as E where EmployeeLastName like 'y%' and T.EmployeeNumber = E.EmployeeNumber)

order by EmployeeNumber

select \*

from tblTransaction as T

Where not exists

(Select EmployeeNumber from tblEmployee as E where EmployeeLastName like 'y%' and T.EmployeeNumber = E.EmployeeNumber)

order by EmployeeNumber

# 10. Top X from various categories

select \* from

(select D.Department, EmployeeNumber, EmployeeFirstName, EmployeeLastName,

rank() over(partition by D.Department order by E.EmployeeNumber) as TheRank

from tblDepartment as D

join tblEmployee as E on D.Department = E.Department) as MyTable

where TheRank <= 5

order by Department, EmployeeNumber

# 11. With Statement

with tblWithRanking as

(select D.Department, EmployeeNumber, EmployeeFirstName, EmployeeLastName,

rank() over(partition by D.Department order by E.EmployeeNumber) as TheRank

from tblDepartment as D

join tblEmployee as E on D.Department = E.Department

select \* from tblWithRanking

where TheRank <= 5

order by Department, EmployeeNumber

with tblWithRanking as

(select D.Department, EmployeeNumber, EmployeeFirstName, EmployeeLastName,

rank() over(partition by D.Department order by E.EmployeeNumber) as TheRank

from tblDepartment as D

join tblEmployee as E on D.Department = E.Department),

Transaction2014 as

(select \* from tblTransaction where DateOfTransaction < '2015-01-01')

select \* from tblWithRanking

left join Transaction2014 on tblWithRanking.EmployeeNumber = Transaction2014.EmployeeNumber

where TheRank <= 5

order by Department, tblWithRanking.EmployeeNumber

# 12. Exercise 1

select E.EmployeeNumber from tblEmployee as E

left join tblTransaction as T

on E.EmployeeNumber = T.EmployeeNumber

where T.EmployeeNumber IS NULL

order by E.EmployeeNumber

select max(EmployeeNumber) from tblTransaction;

with Numbers as (

select top(select max(EmployeeNumber) from tblTransaction) row\_Number() over(order by (select null)) as RowNumber

from tblTransaction as U)

select U.RowNumber from Numbers as U

left join tblTransaction as T

on U.RowNumber = T.EmployeeNumber

where T.EmployeeNumber is null

order by U.RowNumber

select row\_number() over(order by(select null)) from sys.objects O cross join sys.objects P

# 13. Exercise 2

with Numbers as (

select top(select max(EmployeeNumber) from tblTransaction) row\_Number() over(order by (select null)) as RowNumber

from tblTransaction as U),

Transactions2014 as (

select \* from tblTransaction where DateOfTransaction>='2014-01-01' and DateOfTransaction < '2015-01-01'),

tblGap as (

select U.RowNumber,

RowNumber - LAG(RowNumber) over(order by RowNumber) as PreviousRowNumber,

LEAD(RowNumber) over(order by RowNumber) - RowNumber as NextRowNumber,

case when RowNumber - LAG(RowNumber) over(order by RowNumber) = 1 then 0 else 1 end as GroupGap

from Numbers as U

left join Transactions2014 as T

on U.RowNumber = T.EmployeeNumber

where T.EmployeeNumber is null),

tblGroup as (

select \*, sum(GroupGap) over (ORDER BY RowNumber) as TheGroup

from tblGap)

select Min(RowNumber) as StartingEmployeeNumber, Max(RowNumber) as EndingEmployeeNumber,

Max(RowNumber) - Min(RowNumber) + 1 as NumberEmployees

from tblGroup

group by TheGroup

order by TheGroup

# 14. Pivot

with myTable as

(select year(DateOfTransaction) as TheYear, month(DateOfTransaction) as TheMonth, Amount from tblTransaction)

select \* from myTable

PIVOT (sum(Amount) for TheMonth in ([1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12])) as myPvt

ORDER BY TheYear

# 15. Replacing Nulls with Zeros in Pivot

with myTable as

(select year(DateOfTransaction) as TheYear, month(DateOfTransaction) as TheMonth, Amount from tblTransaction)

select TheYear, isnull([1],0) as [1],

isnull([2],0) as [2],

isnull([3],0) as [3],

isnull([4],0) as [4],

isnull([5],0) as [5],

isnull([6],0) as [6],

isnull([7],0) as [7],

isnull([8],0) as [8],

isnull([9],0) as [9],

isnull([10],0) as [10],

isnull([11],0) as [11],

isnull([12],0) as [12] from myTable

PIVOT (sum(Amount) for TheMonth in ([1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12])) as myPvt

ORDER BY TheYear

# 16. UnPivot

SELECT \*

FROM [tblPivot]

UNPIVOT (Amount FOR Month IN ([1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12])) AS tblUnPivot

where Amount <> 0

# 17. Self Joins

begin tran

alter table tblEmployee

add Manager int

go

update tblEmployee

set Manager = ((EmployeeNumber-123)/10)+123

where EmployeeNumber>123

select E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName,

M.EmployeeNumber as ManagerNumber, M.EmployeeFirstName as ManagerFirstName,

M.EmployeeLastName as ManagerLastName

from tblEmployee as E

left JOIN tblEmployee as M

on E.Manager = M.EmployeeNumber

rollback tran

# 18. Recursive CTE

begin tran

alter table tblEmployee

add Manager int

go

update tblEmployee

set Manager = ((EmployeeNumber-123)/10)+123

where EmployeeNumber>123;

with myTable as

(select EmployeeNumber, EmployeeFirstName, EmployeeLastName, 0 as BossLevel --Anchor

from tblEmployee

where Manager is null

UNION ALL --UNION ALL!!

select E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName, myTable.BossLevel + 1 --Recursive

from tblEmployee as E

join myTable on E.Manager = myTable.EmployeeNumber

) --recursive CTE

select \* from myTable

rollback tran

# 19. Scalar Functions 1

CREATE FUNCTION AmountPlusOne(@Amount smallmoney)

RETURNS smallmoney

AS

BEGIN

RETURN @Amount + 1

END

GO

select DateOfTransaction, EmployeeNumber, Amount, dbo.AmountPlusOne(Amount) as AmountAndOne

from tblTransaction

DECLARE @myValue smallmoney

EXEC @myValue = dbo.AmountPlusOne @Amount = 345.67

select @myValue

# 20. Scalar Functions 2

if object\_ID(N'NumberOfTransactions',N'FN') IS NOT NULL

DROP FUNCTION NumberOfTransactions

GO

CREATE FUNCTION NumberOfTransactions(@EmployeeNumber int)

RETURNS int

AS

BEGIN

DECLARE @NumberOfTransactions INT

SELECT @NumberOfTransactions = COUNT(\*) FROM tblTransaction

WHERE EmployeeNumber = @EmployeeNumber

RETURN @NumberOfTransactions

END

# 21. Inline Table Function

CREATE FUNCTION TransactionList(@EmployeeNumber int)

RETURNS TABLE AS RETURN

(

SELECT \* FROM tblTransaction

WHERE EmployeeNumber = @EmployeeNumber

)

SELECT \*

from dbo.TransactionList(123)

select \*

from tblEmployee

where exists(select \* from dbo.TransactionList(EmployeeNumber))

select distinct E.\*

from tblEmployee as E

join tblTransaction as T

on E.EmployeeNumber = T.EmployeeNumber

select \*

from tblEmployee as E

where exists(Select EmployeeNumber from tblTransaction as T where E.EmployeeNumber = T.EmployeeNumber)

# 22. Apply

SELECT \*

from dbo.TransList(123)

GO

select \*, (select count(\*) from dbo.TransList(E.EmployeeNumber)) as NumTransactions

from tblEmployee as E

select \*

from tblEmployee as E

outer apply TransList(E.EmployeeNumber) as T

select \*

from tblEmployee as E

cross apply TransList(E.EmployeeNumber) as T

--123 left join TransList(123)

--124 left join TransList(124)

--outer apply all of tblEmployee, UDF 0+ rows

--cross apply UDF 1+ rows

--outer apply = LEFT JOIN

--cross apply = INNER JOIN

select \*

from tblEmployee as E

where (select count(\*) from dbo.TransList(E.EmployeeNumber)) >3

# 23. Synonyms

create synonym EmployeeTable

for tblEmployee

go

select \* from EmployeeTable

create synonym DateTable

for tblDate

go

select \* from DateTable

create synonym RemoteTable

for OVERTHERE.70-461remote.dbo.tblRemote

go

select \* from RemoteTable

# 24. Dynamic Queries

select \* from tblEmployee where EmployeeNumber = 129;

go

declare @command as varchar(255);

set @command = 'select \* from tblEmployee where EmployeeNumber = 129;'

set @command = 'Select \* from tblTransaction'

execute (@command);

go

declare @command as varchar(255), @param as varchar(50);

set @command = 'select \* from tblEmployee where EmployeeNumber = '

set @param ='129'

execute (@command + @param); --sql injection potential

go

declare @command as nvarchar(255), @param as nvarchar(50);

set @command = N'select \* from tblEmployee where EmployeeNumber = @ProductID'

set @param =N'129'

execute sys.sp\_executesql @statement = @command, @params = N'@ProductID int', @ProductID = @param;

# 25. Problems with IDENTITY

begin tran

insert into tblEmployee2

values ('New Name')

select \* from tblEmployee2

rollback tran

truncate table tblEmployee2

# 26. GUIDs

declare @newvalue as uniqueidentifier --GUID

SET @newvalue = NEWID()

SELECT @newvalue as TheNewID

GO

declare @randomnumbergenerator int = DATEPART(MILLISECOND,SYSDATETIME())+1000\*(DATEPART(SECOND,SYSDATETIME())

+60\*(DATEPART(MINUTE,SYSDATETIME())+60\*DATEPART(HOUR,SYSDATETIME())))

SELECT RAND(@randomnumbergenerator) as RandomNumber;

begin tran

Create table tblEmployee4

(UniqueID uniqueidentifier CONSTRAINT df\_tblEmployee4\_UniqueID DEFAULT NEWID(),

EmployeeNumber int CONSTRAINT uq\_tblEmployee4\_EmployeeNumber UNIQUE)

Insert into tblEmployee4(EmployeeNumber)

VALUES (1), (2), (3)

select \* from tblEmployee4

rollback tran

go

declare @newvalue as uniqueidentifier

SET @newvalue = NEWSEQUENTIALID()

SELECT @newvalue as TheNewID

GO

begin tran

Create table tblEmployee4

(UniqueID uniqueidentifier CONSTRAINT df\_tblEmployee4\_UniqueID DEFAULT NEWSEQUENTIALID(),

EmployeeNumber int CONSTRAINT uq\_tblEmployee4\_EmployeeNumber UNIQUE)

Insert into tblEmployee4(EmployeeNumber)

VALUES (1), (2), (3)

select \* from tblEmployee4

rollback tran

# 27. Defining SEQUENCES

BEGIN TRAN

CREATE SEQUENCE newSeq AS BIGINT

START WITH 1

INCREMENT BY 1

MINVALUE 1

--MAXVALUE 999999

--CYCLE

CACHE 50

CREATE SEQUENCE secondSeq AS INT

SELECT \* FROM sys.sequences

ROLLBACK TRAN

# 28. NEXT VALUE FOR sequence

BEGIN TRAN

CREATE SEQUENCE newSeq AS BIGINT

START WITH 1

INCREMENT BY 1

MINVALUE 1

CACHE 50

select NEXT VALUE FOR newSeq as NextValue;

--select \*, NEXT VALUE FOR newSeq OVER (ORDER BY DateOfTransaction) as NextNumber from tblTransaction

rollback tran

CREATE SEQUENCE newSeq AS BIGINT

START WITH 1

INCREMENT BY 1

MINVALUE 1

--MAXVALUE 999999

--CYCLE

CACHE 50

alter table tblTransaction

ADD NextNumber int CONSTRAINT DF\_Transaction DEFAULT NEXT VALUE FOR newSeq

alter table tblTransaction

drop DF\_Transaction

alter table tblTransaction

drop column NextNumber

alter table tblTransaction

add NextNumber int

alter table tblTransaction

add CONSTRAINT DF\_Transaction DEFAULT NEXT VALUE FOR newSeq for NextNumber

begin tran

select \* from tblTransaction

INSERT INTO tblTransaction(Amount, DateOfTransaction, EmployeeNumber)

VALUES (1,'2017-01-01',123)

select \* from tblTransaction WHERE EmployeeNumber = 123;

update tblTransaction

set NextNumber = NEXT VALUE FOR newSeq

where NextNumber is null

select \* from tblTransaction --WHERE EmployeeNumber = 123

ROLLBACK TRAN

--SET IDENTITY\_INSERT tablename ON

--DBCC CHECKIDENT(tablename,RESEED)

alter sequence newSeq

restart with 1

alter table tblTransaction

drop DF\_Transaction

alter table tblTransaction

drop column NextNumber

DROP SEQUENCE newSeq

# 31. Introducing XML

declare @x xml

set @x = '<Shopping ShopperName="Phillip Burton" Weather="Nice">

<ShoppingTrip ShoppingTripID="L1">

<Item Cost="5">Bananas</Item>

<Item Cost="4">Apples</Item>

<Item Cost="3">Cherries</Item>

</ShoppingTrip>

<ShoppingTrip ShoppingTripID="L2">

<Item>Emeralds</Item>

<Item>Diamonds</Item>

<Item>Furniture</Item>

</ShoppingTrip>

</Shopping>'

select @x

update [dbo].[tblEmployee]

set XMLOutput = @x

where EmployeeNumber = 200

select \* from [dbo].[tblEmployee]

# 32. FOR XML RAW

select E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName

, E.DateOfBirth, T.Amount, T.DateOfTransaction

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeNumber between 200 and 202

for xml raw('MyRow'), elements

# 33. FOR XML AUTO

select E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName

, E.DateOfBirth, T.Amount, T.DateOfTransaction

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeNumber between 200 and 202

for xml auto, elements

# 34. FOR XML PATH

select E.EmployeeFirstName as '@EmployeeFirstName'

, E.EmployeeLastName as '@EmployeeLastName'

, E.EmployeeNumber

, E.DateOfBirth

, T.Amount as 'Transaction/Amount'

, T.DateOfTransaction as 'Transaction/DateOfTransaction#'

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeNumber between 200 and 202

for xml path('Employees'), ROOT('MyXML')

# 35. FOR XML EXPLICIT

select 1 as Tag, NULL as Parent

, E.EmployeeFirstName as [Elements!1!EmployeeFirstName]

, E.EmployeeLastName as [Elements!1!EmployeeLastName]

, E.EmployeeNumber as [Elements!1!EmployeeNumber]

, E.DateOfBirth as [Elements!1!DateOfBirth]

, null as [Elements!2!Amount]

, null as [Elements!2!DateOfTransaction]

from [dbo].[tblEmployee] as E

where E.EmployeeNumber between 200 and 202

union all

select 2 as Tag, 1 as Parent

, null as [EmployeeFirstName]

, null as [EmployeeLastName]

, T.EmployeeNumber

, null as DateOfBirth

, Amount

, DateOfTransaction

from [dbo].[tblTransaction] as T

inner join [dbo].[tblEmployee] as E on T.EmployeeNumber = E.EmployeeNumber

where T.EmployeeNumber between 200 and 202

order by EmployeeNumber, [Elements!2!Amount]

for xml explicit

# 35. XQuery Value method

declare @x xml

set @x='<Shopping ShopperName="Phillip Burton" >

<ShoppingTrip ShoppingTripID="L1" >

<Item Cost="5">Bananas</Item>

<Item Cost="4">Apples</Item>

<Item Cost="3">Cherries</Item>

</ShoppingTrip>

<ShoppingTrip ShoppingTripID="L2" >

<Item>Emeralds</Item>

<Item>Diamonds</Item>

<Item>Furniture</Item>

</ShoppingTrip>

</Shopping>'

select @x.value('(/Shopping/ShoppingTrip/Item/@Cost)[1]','varchar(50)')

# 36. XQuery Modify method

set @x.modify('replace value of (/Shopping/ShoppingTrip[1]/Item[3]/@Cost)[1]

with "6.0"')

select @x

set @x.modify('insert <Item Cost="5">New Food</Item>

into (/Shopping/ShoppingTrip)[2]')

select @x

# 37. XQuery Query and FLWOR 1

select @x.query('for $ValueRetrieved in /Shopping/ShoppingTrip/Item

return $ValueRetrieved')

select @x.query('for $ValueRetrieved in /Shopping/ShoppingTrip/Item

return string($ValueRetrieved)')

select @x.query('for $ValueRetrieved in /Shopping/ShoppingTrip[1]/Item

return concat(string($ValueRetrieved),";")')

# 38. XQuery Query and FLWOR 2

select @x.query('for $ValueRetrieved in /Shopping/ShoppingTrip[1]/Item

let $CostVariable := $ValueRetrieved/@Cost

where $CostVariable >= 4

order by $CostVariable

return concat(string($ValueRetrieved),";")')

# 39. nodes using Variable (shredding a variable)

select tbl.col.value('.', 'varchar(50)') as Item

, tbl.col.value('@Cost','varchar(50)') as Cost

into tblTemp

from @x.nodes('/Shopping/ShoppingTrip/Item') as tbl(col)

select \* from tblTemp

drop table tblTemp

--for let where order by return

# 40. notes using table (shredding a table)

declare @x1 xml, @x2 xml

set @x1='<Shopping ShopperName="Phillip Burton" >

<ShoppingTrip ShoppingTripID="L1" >

<Item Cost="5">Bananas</Item>

<Item Cost="4">Apples</Item>

<Item Cost="3">Cherries</Item>

</ShoppingTrip></Shopping>'

set @x2='<Shopping ShopperName="Phillip Burton" >

<ShoppingTrip ShoppingTripID="L2" >

<Item>Emeralds</Item>

<Item>Diamonds</Item>

<Item>Furniture</Item>

</ShoppingTrip>

</Shopping>'

drop table #tblXML

create table #tblXML(pkXML INT PRIMARY KEY, xmlCol XML)

insert into #tblXML(pkXML, xmlCol) VALUES (1, @x1)

insert into #tblXML(pkXML, xmlCol) VALUES (2, @x2)

select \* from #tblXML

select tbl.col.value('@Cost','varchar(50)')

from #tblXML CROSS APPLY

xmlCol.nodes('/Shopping/ShoppingTrip/Item') as tbl(col)

# 41. Importing and exporting XML using the bcp utility

bcp [70-461S5].dbo.tblDepartment out mydata.out -N -T

create table dbo.tblDepartment2

([Department] varchar(19) null,

[DepartmentHead] varchar(19) null)

bcp [70-461S5].dbo.tblDepartment2 in mydata.out -N –T

# 42. Bulk Insert and Openrowset

drop table #tblXML

go

create table #tblXML(XmlCol xml)

go

bulk insert #tblXML from 'C:\XML\SampleDataBulkInsert.txt'

select \* from #tblXML

drop table #tblXML

go

create table #tblXML(IntCol int, XmlCol xml)

go

insert into #tblXML(XmlCol)

select \* from

openrowset(BULK 'C:\XML\SampleDataOpenRowset.txt', SINGLE\_BLOB) AS x

select \* from #tblXML

# 43. Schema

select E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName

, T.Amount, T.DateOfTransaction

from [dbo].[tblEmployee] as E

left join [dbo].[tblTransaction] as T

on E.EmployeeNumber = T.EmployeeNumber

where E.EmployeeNumber between 200 and 202

for xml raw, xmlschema --, xmldata

|  |  |  |
| --- | --- | --- |
| i4 or int | Int | Whole number, [integer](https://en.wikipedia.org/wiki/Integer) |
| Boolean |  | [Boolean](https://en.wikipedia.org/wiki/Boolean_datatype) logical value (0 or 1) |
| dateTime.iso8601 | Datetime | Date and time in [ISO 8601](https://en.wikipedia.org/wiki/ISO_8601)format |
| Double |  | [Double precision](https://en.wikipedia.org/wiki/Double_precision) floating point number |
| String | Varchar | String of characters. Must follow [XML encoding](https://en.wikipedia.org/wiki/XML#Characters_and_escaping). |
| Nil | Null | [Discriminated null value](https://en.wikipedia.org/wiki/Nullable_type); an XML-RPC [extension](https://web.archive.org/web/20050911054235/http:/ontosys.com/xml-rpc/extensions.php) |

# 46. XML Indexes

declare @x1 xml, @x2 xml

set @x1='<Shopping ShopperName="Phillip Burton" >

<ShoppingTrip ShoppingTripID="L1" >

<Item Cost="5">Bananas</Item>

<Item Cost="4">Apples</Item>

<Item Cost="3">Cherries</Item>

</ShoppingTrip></Shopping>'

set @x2='<Shopping ShopperName="Phillip Burton" >

<ShoppingTrip ShoppingTripID="L2" >

<Item>Emeralds</Item>

<Item>Diamonds</Item>

<Item>Furniture

<Color></Color></Item>

</ShoppingTrip>

</Shopping>'

drop table #tblXML;

create table #tblXML(pkXML INT PRIMARY KEY, xmlCol XML)

insert into #tblXML(pkXML, xmlCol) VALUES (1, @x1)

insert into #tblXML(pkXML, xmlCol) VALUES (2, @x2)

create primary xml index pk\_tblXML on #tblXML(xmlCol)

create xml index secpk\_tblXML\_Path on #tblXML(xmlCol)

using xml index pk\_tblXML FOR PATH

create xml index secpk\_tblXML\_Value on #tblXML(xmlCol)

using xml index pk\_tblXML FOR VALUE

create xml index secpk\_tblXML\_Property on #tblXML(xmlCol)

using xml index pk\_tblXML FOR PROPERTY