**Create** **database** bookstore;

Use bookstore

**Create table** books

(

Book\_id varchar(50),

Title varchar(50),

Category varchar(50),

Author varchar(50),

Price double

);

Right click table to

create table,

select top 100 rows

script table as

**Desc** books;

Datatype –

Integer whole num

Varchar 255 text

Blog

Decimal

Date

Datetime

**Insert into** books

**Values**

(4,’only time will tell’,’fiction’,’Jeffery’,18.78)

--**Adding different sequence**

Insert into books

(author, price, category, title, book\_id)

Values

(‘Jeffary’, 18.78,’fiction’,’only time will tell’,4)

--adding partial values

Insert into books(book\_id,Author,title)

Values (4,’Jeffery’,’Only time will tell’)

-- inserts null values in rest

-- Null – missing values

-- Null not equal 0/empty string

-- two nulls can not be compared

-- Can restrict columns to accept null

**Install Adventure works database**

Download – unzip

- mdf – actual database file

- ldf – actual log file

Right click on database / Attach / path of the mdf file

**Atomic Data**

Rule 1:

Column should not have several values of same DT

|  |  |
| --- | --- |
| **Food** | **Ingredients** |
| Salad | Tomato, cucumber, onion |
| Tea | Milk, Tea leaves, Sugar |

|  |  |  |  |
| --- | --- | --- | --- |
| **Food** | **Ingredient 1** | **Ingredient 2** | **Ingredient 3** |
| Salad | Tomato | Cucumber | Onion |
| Tea | Milk | Tea leaves | Sugar |

Rule 2:

Table should not have multiple columns with same type of data

|  |  |
| --- | --- |
| **Food\_id** | **Tomato** |
| F01 | Cucumber |
| F02 | Milk |
| F02 | Tea leaves |
| F01 | Onion |
| F02 | Sugar |
| F01 | Tomato |

Atomicity Reached – Normal Form 1

**Normalization** – Set or Rules to design table

* Reduce size
* Faster query
* Remove duplicate

**1 NF**

Rule 1: Each row has Atomic value

Rule 2: Each row must have unique id (PK)

* PK cannot be null
* Unique
* Cannot changed

**Composite key** – 2/more columns uniquely identify

Functional dependency

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Book | Category | Customer | Author | Copyright |
|  |  |  |  |  |

Book – Author, category, copyright, customer

**Partial Functional dependency**

**Composite key** column – Book, Customer

Non key Column –

When non key column depends only on some column (not all) of composite key.

**Transitive Functional Key**

|  |  |  |  |
| --- | --- | --- | --- |
| Book | Category | Author | Phone |
|  |  |  |  |

Book – pk

Other – non key

Change in Author changes Phone number

Change in non key column impacts other non key column – Transitive Functional Key

**2NF**

Rule 1: 1NF

Rule 2: Table should not have any partial functional dependencies

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Book | Customer | Category | Author | Phone |
|  |  |  |  |  |

Book\_info

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| B\_id | Book | Category | Author | copyright |
|  |  |  |  |  |

Customer\_info

|  |  |  |  |
| --- | --- | --- | --- |
| C\_id | Customer | Phone | B\_id(FK) |
|  |  |  |  |

**Foreign Key**

* Ref PK of other table
* It can have different name
* Do not have to be unique
* Can be null
* Restrict to not null – for join

**3NF**

Rule 1: 2NF

Rule 2: Table should not have transitive dependency

**Transitive dependency**

Courses

|  |  |  |
| --- | --- | --- |
| course | instructor | Phone |
|  |  |  |

Courses

|  |  |
| --- | --- |
| Course\_id | Course |
|  |  |

Instructor

|  |  |  |  |
| --- | --- | --- | --- |
| Instructor\_id | Instructor | Phone | Course\_id(FK) |
|  |  |  |  |

**Schema**



Select \* from books

Select \* from books

Where author = ‘Jeffrey Archer’’

Select title, author

from books

Where author = ‘Jeffrey Archer’’;

Select \*

from books

Where author = ‘Jeffrey Archer’’

**And** category = ‘Fiction’

And Price > 19;

**Comparison operator**

=

<>

<

>

<=

>=

Can use with text data:

Select book\_id, title, author from books

where **title >** ‘T’

Select \*

from books

Where category = ‘Fiction’

**OR** category = ‘Biography’

Select \*

from books

Where address like ‘%CA’

Wildcard character

% macca, jamaica

\_ BCA, CCA

**Range Selection**

Select title, price from books

Where price between 18 and 22

Select title from books

Where authors in (‘Jeffery Archer’,’Colinho’)

Select title from books

Where author not in (‘Jeffery Archer’)

Select title, price from books

Where NOT price between 18 and 22

Select \* from books

Where NOT address like ‘%CA’

**Ordering**

Select bookId, title, price From books

Where category = ‘literature

Order by title

Select \* from books

Order by purchased desc;

**SQL Functions**

Select sum(sales) from book\_sale

Where name = ‘Jim’

Select name, sum(sales) from book\_sale

**Group by** name

**Order by** sum(sales)

Select name, avg(sales) from book\_sale

**Group by** name

Select name, max(sales) from book\_sale

Group by name

Select name, min(sales) from book\_sale

Group by name

Select count(sales\_date) from book\_sale

Select count(distinct(sales\_date) )

from book\_sale

select name, sum(sales) from book\_sales

group by name

order by sum(sales) limit 2;

**Adventure works database**

* Select \* from [].[] drag-drop
* Select names from
* Select distinct grp\_names from
* Select \* from [].[] --exact match, not case sensitive

Where grpnames like ‘manufacturing’

* Select \* from [HumanResources].[Employees]

Where organization level = 2

* Select \* from [HumanResources].[Employees]

Where organization level in (2,3)

* Select \* from [].[]

Where title like ‘%Manager’

* Select \* from [HumanResources].[Employees]

Where birthdate > ‘1/1/1980’

* Select \* from [HumanResources].[Employees]

Where birthdate between ‘1/1/1970’ and ‘1/1/1980’

* Select name, listprice, listprice + 10 as adj\_price from [].[]
* Select name, listprice, listprice + 10 as adj\_price

Into [].[\_2]

from [].[]

* Select name, listprice, listprice + 10 as adj\_price

Into #tempname

from [].[]

* Delete from [].[] Where name like ‘ ‘
* Update [].[]

Set Name = ‘’

Where name like ‘’

* Select a.name,b.name,b.sal

From emp a inner join salary b on a.empid = b.empid

Left join

Right join

Full outer join

Cross join –or select \* from tab1,tab2

* Select getdate()
* Select getdate() – 2
* Select datepart(yyyy, getdate())
* Select datepart(mm, getdate())
* Select dateadd(day, 4, getdate())
* Select dateadd(month,4,’7/4/2015’)
* Select workordeid, datediff(day,startdate,enddate)

From [].[]

* select avg(sal) from tab
* select count(salary) from tab
* select count(\*) from tab

sum, min, max

* select concat(‘f\_name’,’ ’,’lname’) from emp
* select left(ordername,5) from myorder

--first 5 from left side

Right(column,5)

Substring(column,2,5)

Lower(column)

Upper(column)

Len(column)

* Concat (Upper(left(column,1)),

lower(substring(column,2,len(column)))

Ltrim(column) –remove white spaces from left

Rtrim(column)

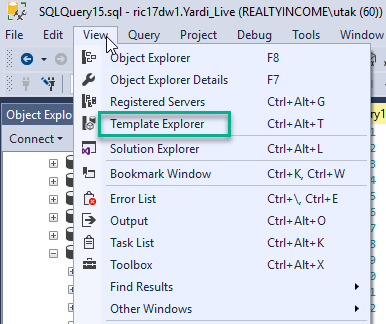
Ltrim(rtrim(column))

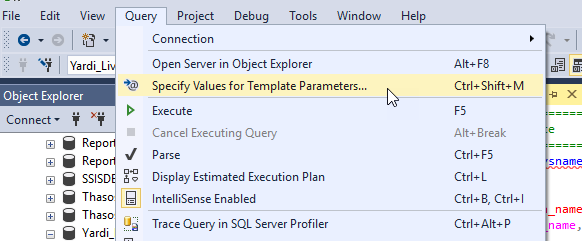
**Views**

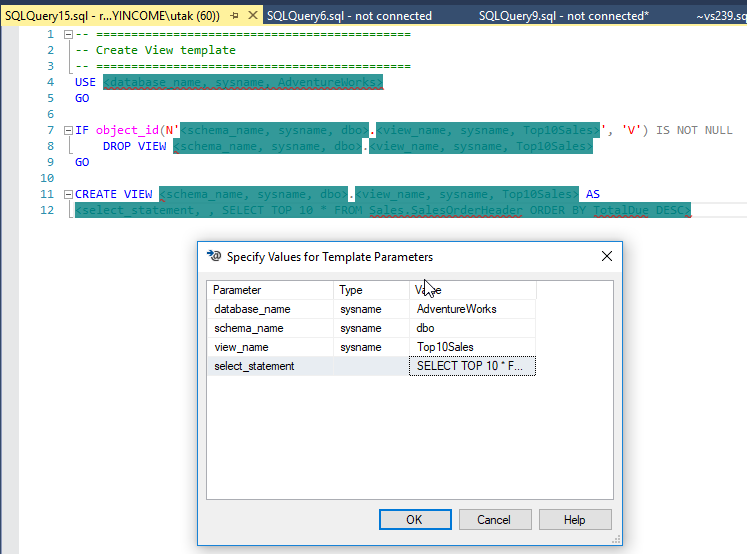
Restricted access – opens a window for table

Multiple windows to access certain columns of same table

1. Rt click – new view – select table – column
2. Template explorer in file ribbon / view – template browser – create view - Qury ribbon – specify values for template parameters – replace template







Create view MyCustomUSview

as

Select \* from [].[] Where

**Triggers**

Statement that run when Some event happens in table.

Insert/delete/update

$1000 sale – Email

$1M sale – rollback

1. Rt click / new trigger
2. Template Explorer

---Statement level trigger

Create trigger name

On [].[]

After insert

As

Begin

Print ‘insert is not allowed’

Rollback transaction

End

go

--database level trigger

Create trigger name

On database

After create\_table

As

Begin

Print ‘creation of new table not allowed’

Rollback transaction

End

go

**SQL Computed columns**

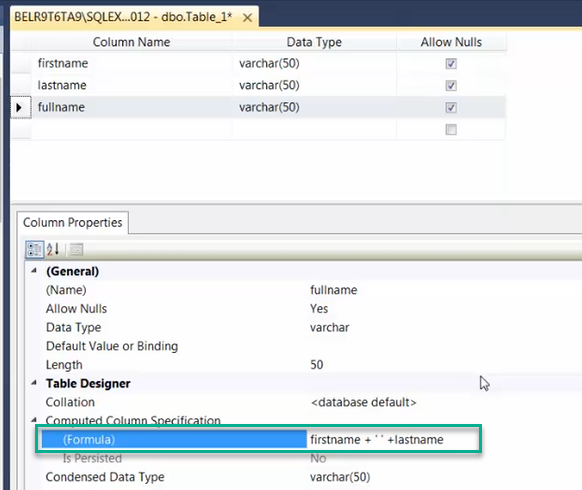
Fname, lname

Computed column – full name

GUI based approach

1. Create table

Fname, lname, fullname



**Stored procedure**

Shopping website

Search shoes – list of shoes

1. Retrieve information with bunch of queries - SP

Get\_all\_prod(prod\_name) –parameterized SP

Parameter as filtering products

Stored in table backend

1. Add to cart

Insert in cart table

Update to cart if changes

Delete f item is removed

Calling pre written stored procedure

Reusable

Centralize logic

SP created – cashed – execution plan is created

1. GUI method – Right click
2. Template method
3. Write from scratch

**Create** procedure ProcedureName

As

Set nocount on/off

Select \* from [].[]

**Exec** ProcedureName

**Drop** proc ProcedureName

**Parameterized procedure**

Create procedure MyFirstParamproc

@param\_name varchar(50)

As

Set nocount on

Select \* from [].[]

Where name = @ param\_name

EXEC MyFirstParamproc @Param\_Name = ‘Day’

EXEC MyFirstParamproc –error without param

**Default paramaeter**

Create procedure MyFirstParamproc

@param\_name varchar(50) = ‘Evening’

As

Set nocount on

Select \* from [].[]

Where name = @ param\_name

**Output Parameter**

Create proc MyOutputSP

@topShift varchar(50) OUTPUT

AS

SET @topShift = (select top 1 col fom tabl)

Declare @outputresult varchar(50)

Exec MyOutputSP @outputresult OUTPUT

Select @outputresult

**Returning value from SP**

Create proc nameProc

As

Return 12

Declare @returnval INT

Exec @returnval = nameProc

Select @returnval

**User defined Functions (UDF)**

Return single scalar value or result set

Can be inline

1. Scalar function
2. Table valued function
3. System functions – getdate()

**Scalar functions**

Create function YTDSALES()

Return Money

As

Begin

Declare @ytdsales money

Select @ytdsales = sum(salesytd) from [].[]

Return @ytdsales

END

DECLARE @YTDRESULTS AS MONEY

SELECT @YTDRESULTS = dbo. YTDSALES()

PRINT @YTDRESULTS

**DROP FUNCTION** YTDSALES

**Parameterized function**

Create function YTD\_GROUP(@Group varchar(50))

RETURNS MONEY

AS

BEGIN

DECLARE @YTDSALES AS MONEY

Select @YTDSALES = sum(sales) from [].[]

Where col = @ Group

Return @YTDSALES

END

Declare @RESULTS MONEY

Select @RESULTS dbo.YTD\_GROUP(‘North America’)

Print @RESULTS

**Function returning table**

Create function Name\_tabvalued (@TerritoryID INT)

RETURNS TABLE

AS RETURN

Select col1, col2, col3 from [].[]

Where col = @TerritoryID

Select col1 from dbo. Name\_tabvalued(7)

**SQL Transactions and error handling**

Store database changes only if

all the transaction run successfully,

no partial success

ACID – atomicity, consistency, isolation, durability

**Atomicity** – all or nothing

**Consistency** – ensures transaction will bring database from one valid state to another

**Isolation** – Concurrent exec of transaction result in a system state that would be obtained if transactions were executed serially

by table level lock, row level lock even though running parallelly

**Durability** – once stored in data it should remain in database**.** Data should be persistent.

**Try** {error}

**Catch** {restore logic}

**Transitions**

Begin transaction

Update [].[]

Set column = 1

Where col2 = 1

COMMIT TRANSACTION

--@@error

0 = success , >0 error

Declare @errorresult varchar(50)

Begin TRANSACTION

INSERT INTO []. []

(col1, col2, col3)

Values (val1, val2, val3)

Set @errorresult = @@ERROR

If (@errorresult = 0)

BEGIN

Print ‘Success!’

COMMIT TRANSACTION

END

Else

BEGIN

Print ‘transaction Failed’

ROLLBACK TRANSACTION

END

**Custom Error Message**

BEGIN

Print ‘Success!’

COMMIT TRANSACTION

END

Else

BEGIN

RAISERROR(‘Statement Failed – This is my custom message’ , 16, 1)

ROLLBACK TRANSACTION

END

**TRY-CATCH**

BEGIN TRY

BEGIN TRANSACTION

INSERT INTO []. [] (col1, col2, col3)

VALUES (val1, val2, val3)

COMMIT TRANSACTION

END TRY

BEGIN CATCH

Print ‘Catch Statement Entered’

Rollback transaction

END CATCH

Deadlocks – r1 waiting for r2, r2 waiting for r1

Deadlock graph

– dead resolving mechanism

* Who is the victim

**Common table expressions (CTE)**

Store temporary –

WITH CTE\_SALENAME

AS

(

Select col1, col2 from [].[]

)

Select \* from CTE\_SALENAME

WHERE NAME LIKE ‘North%’

Can be Self recursive

Break down complex query

Join CTE is possible

**Grouping Sets**

Select \* from [].[]

Select name, Sum(sales)

From [].[]

Group By Name

Select Name,Country,col3 Sum(sales)

From [].[]

Group By Name, country, col3

--error – column dt mismatch

Select Name, Sum(sales)

From [].[]

Group By Name

Union all

Select Name, col2, Sum(sales)

From [].[]

Group By Name, col2

Union all

Select Name, col2, col3, Sum(sales)

From [].[]

Group By Name, col2, col3

Alternative

Select Name, null, null Sum(sales)

From [].[]

Group By Name

Union all

Select Name, col2, null Sum(sales)

From [].[]

Group By Name, col2

Union all

Select Name, col2, col3, Sum(sales)

From [].[]

Group By Name, col2, col3

**Grouping sets**

--similar and easier

Select Name, country, company, Sum(sales)

From [].[]

Group By GROUPING SETS

(

(Name)

(Name, country)

(Name, country, company)

)

**Rollup**

--Similar and easier rollup

Select Name, country, company, Sum(sales)

From [].[]

Group By ROLLUP

(

(Name, country, company)

)

**CUBE**

All combinations of column Name, Country, Company

Select Name, country, company, Sum(sales)

From [].[]

Group By CUBE

(

(Name, country, company)

)

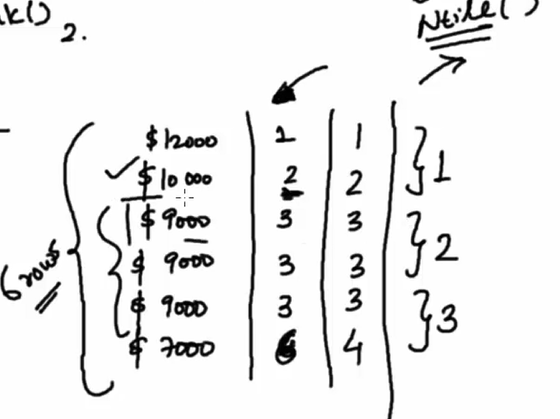
**Ranks**

Eid, Name, sal –order by sal, ranking based on ordering

Rank(), dense rank()

Bucketing

After ordering – putting in bucket – Ntile



**ROW\_NUMBER()**

Select

postal\_code,

,ROW\_NUMBER() OVER (ORDER BY postal\_code) as ‘rownum’

from address

Where postal\_code in (,,,,)

**RANK()** –1,1,1,4

Select

postal\_code

,ROW\_NUMBER() OVER (ORDER BY postal\_code) as ‘rownum’

,RANK() OVER (ORDER BY postal\_code) as ‘rank’

from address

Where postal\_code in (,,,,)

**DENSE\_RANK()** –1,1,1,2

Select

postal\_code

,ROW\_NUMBER() OVER (ORDER BY postal\_code) as ‘rownum’

,RANK() OVER (ORDER BY postal\_code) as ‘rank’

,DENSE\_RANK() OVER (ORDER BY postal\_code)

as ‘dense rank’

from address

Where postal\_code in (,,,,)

**NTILE()** —bucketing in group

—in this case total 10 groups46/451

Select

postal\_code

,ROW\_NUMBER() OVER (ORDER BY postal\_code) as ‘rownum’

,RANK() OVER (ORDER BY postal\_code) as ‘rank’

,DENSE\_RANK() OVER (ORDER BY postal\_code)

as ‘dense rank’

,NTILE(10) OVER (ORDER BY postal\_code) as ‘ntile’

from address

Where postal\_code in (,,,,)

**SQL XML Datatype**

Email – to, from, cc, bcc, subject, message

Tags <to></to>,<from></from>

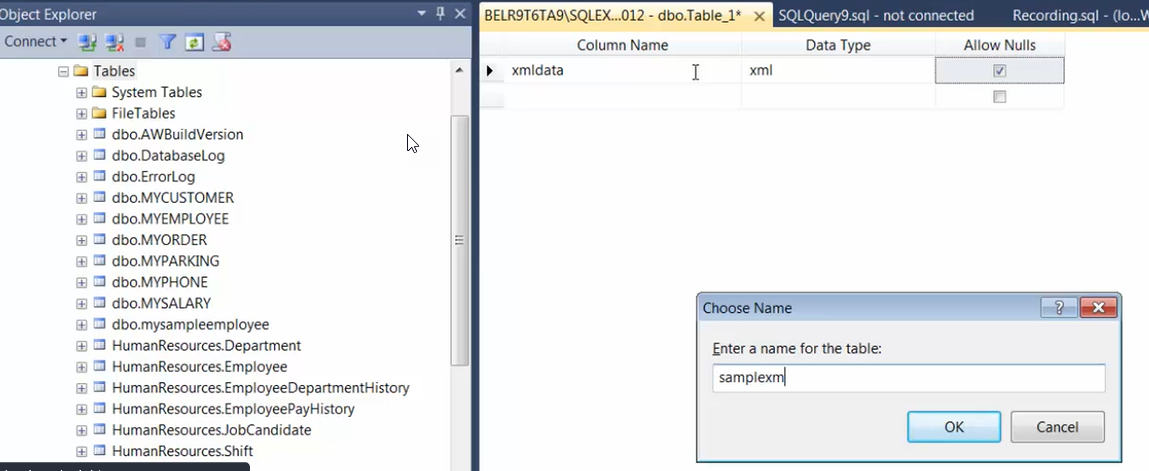
text file - schema definition

xslt -> xml transfer to html

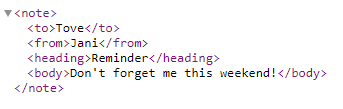
parse – xpath = inside message/inside to

for each syntax

**XML DATATYPE**



Select \* from samplexmltable



Insert into [dbo].[samplexmltable] (xmldata)

Values(

‘<note>

<to>Tove</to>

<from>Jani</from>

<heading>Reminder</heading>

<body>Dont forget me this weekend!</body>

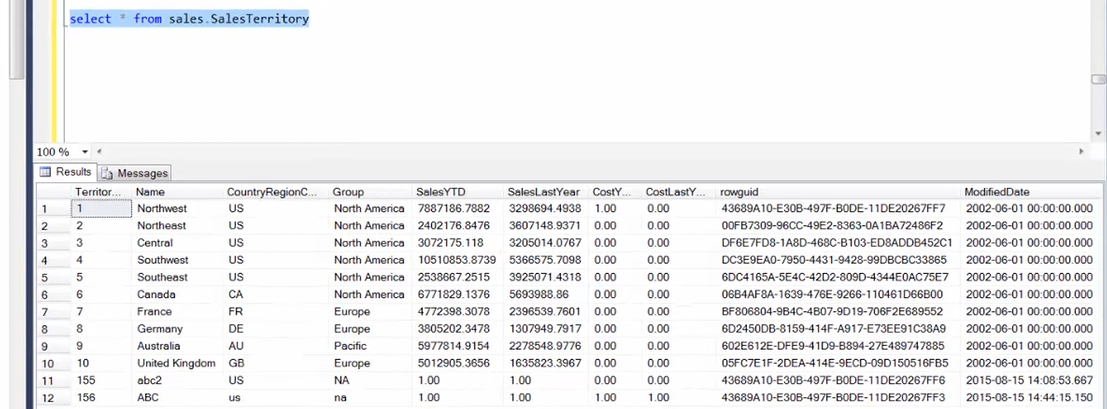
</note>’

)



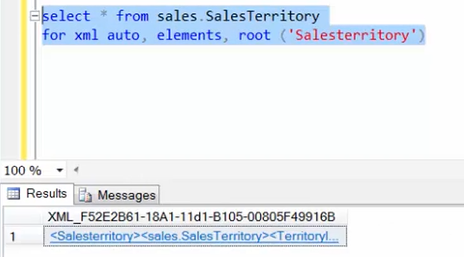
Link click – show data in actual format

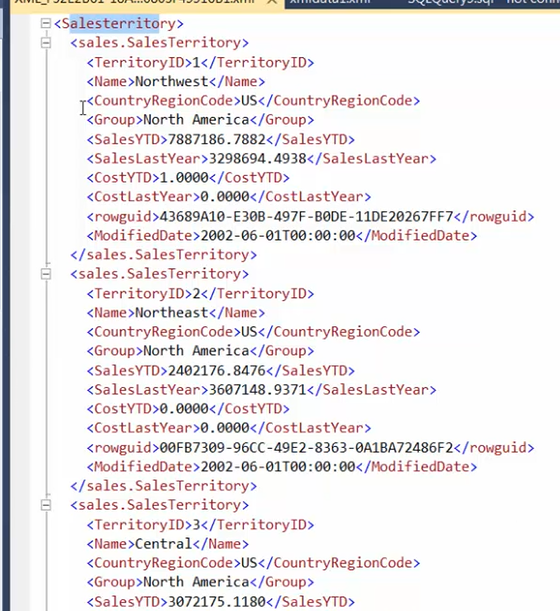
Convert normal table into xml



Select \* from table

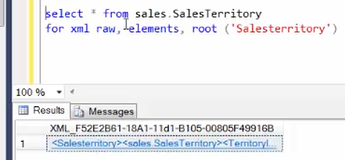
For xml auto, elements, root(‘Salesterritory’)

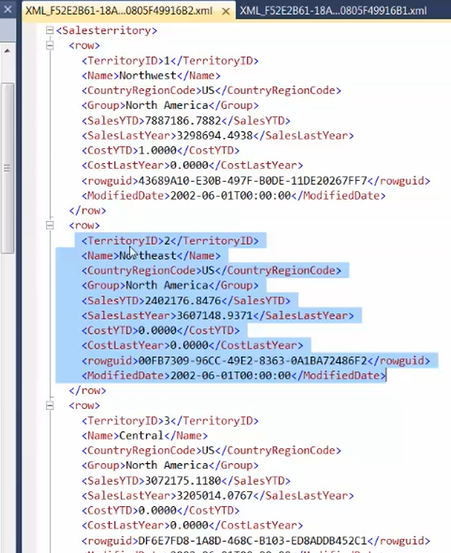




Select \* from table

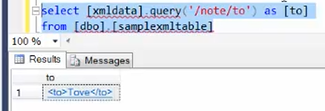
For xml raw, elements, root(‘Salesterritory’)





Select [xmldata].query(‘/note/to’) as [to]

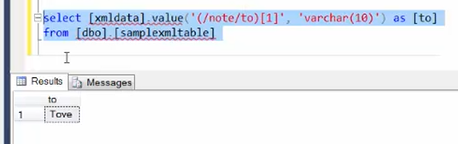
From [dbo].[sampleexmltable]



Select [xmldata].value(‘(/note/to)[1]’,’’varchar(10)’) as [to]

From [dbo].[sampleexmltable]

--get the first value, define size



--top 10 id in xml format

Select top 10 territoryid from salesTerritory

For xml auto, element, root(‘SalesTerritory’)

--parsing xml to table format

Declare @xmlhandle int

Declare @xmldocument xml

Set @xmldocument =

(

Select \* from SalesTerritory

For xml auto, elemenets, root(‘SalesTerritory’)

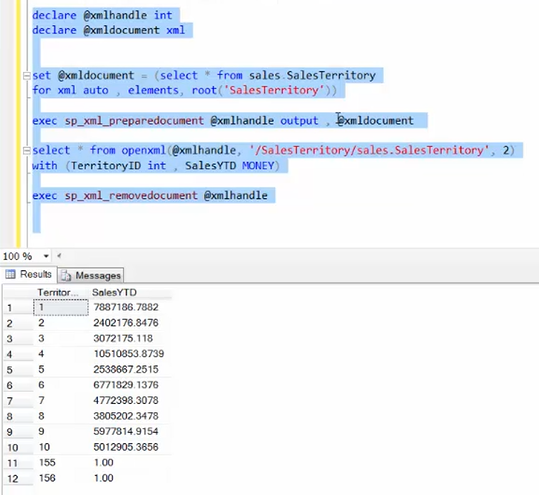
)

Exec sp\_xml\_preparedocument @xmlhandle output, @xmldocument

Select \* from openxml(@xmlhandle, ‘/SalesTerritory/sales.SalesTerritory’, 2) –go 2 level down

With (TerritoryID INT, Sales YTD MONEY)

Exec sp\_xml\_removedocument @xmlhandle



**SQL Partitions**

Database – insert and select – 5M rows – time consuming

Table can be partitioned - <2009, 2009-2010, >2010

Limiting scope

Increasing speed

Breaking table in smaller pieces

Need enterprise edition

New db

Create partition function cust\_part\_func(int)

As range right

For values (1000, 2000, 3000, 4000, 5000)

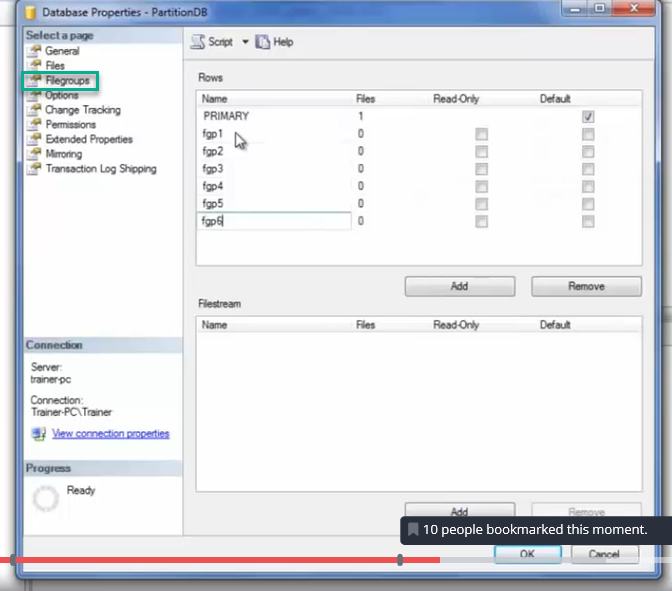
--create 5 buckets, declared exclusive syntax

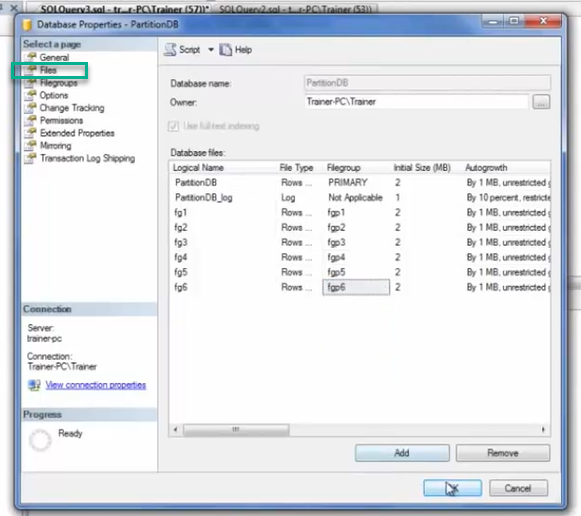
Create partition scheme cust\_part\_scheme

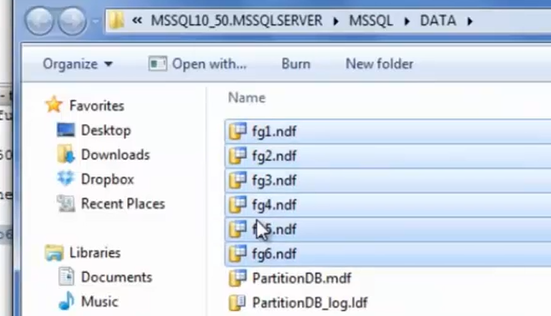
As partition cust\_part\_func

To (fgp1 ,fgp2, fgp3, fgp4, fgp5, fgp6)

Database properties – Partition – adding file groups







Create table partition

(

Eid int identity (1,1) not null,

Empdate datetime null

)

On cust\_part\_scheme (empid)

Declare @i int

Set @i = 0

While @i <10000

Begin

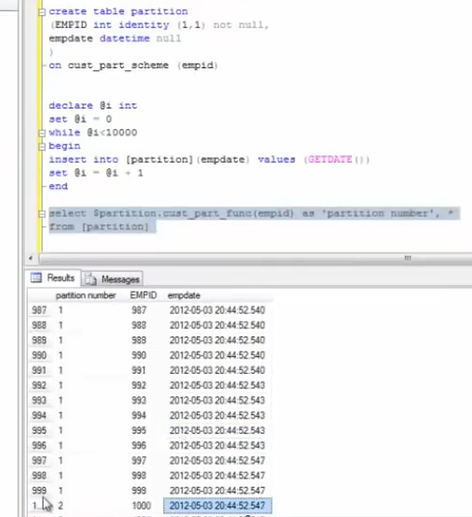
Insert into [partition](empdate) values (getdate())

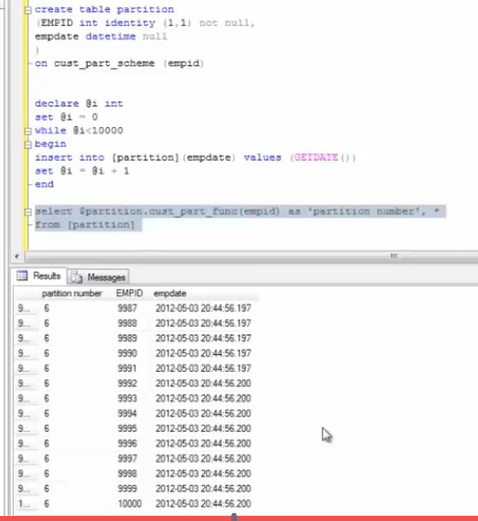
Set @i = @i+1

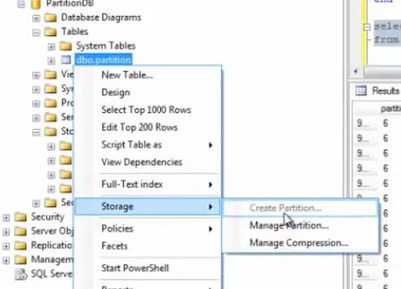
End

Select @partition.cust\_part\_func(empid) as ‘partition num’

From [partition]







**Pivot**



Dynamically created

Table names are not known unless created as not aware about what data it has

Select \* from salesterritory

Select region, [group], salesytd

From salesterritory

Countrycode | North America | Europe

US | 23 | ----

Select region, [North America], [Europe]

From salesterritory

Pivot

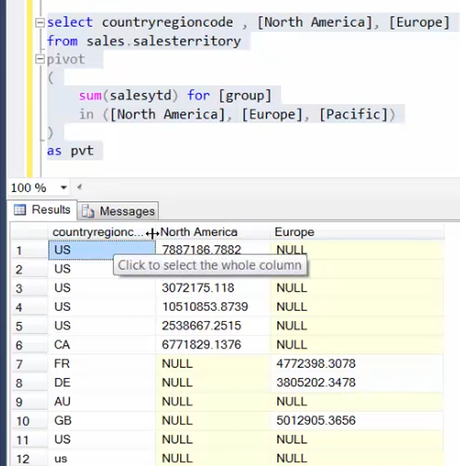
(

Sum(salesytd) for [group]

In([North America], [Europe], [Pacific])

)

As pvt



**SQL Dynamic Queries**

Create string dynamically

And execute once it is complete

@var1 = na, eur

@var2 = select \* from

@sql = @var1+ @var2

Declare @sqlstr varchar(2000)

Set @sglstr = ‘select countryregion, [group], ’

Set @sqlstring = @sqlstring + ‘salesytd from salesterritory’

Print @sqlstring

Exec (@sqlstring)

Filestream functionality

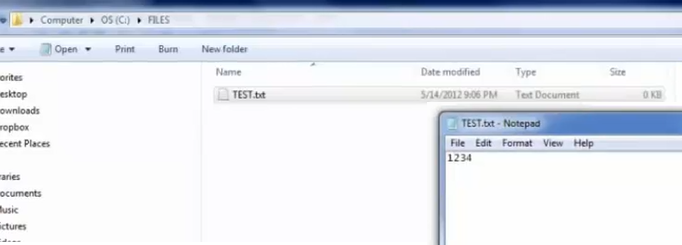
Create table files

(

Docid uniqueidentifier notnull rowguidcol unique default newid()

Doc varbinary (max) filestream null

)



Declare @doc as varbinary (max)

Select @doc = cast(bulkcolumn as varbinary(max))

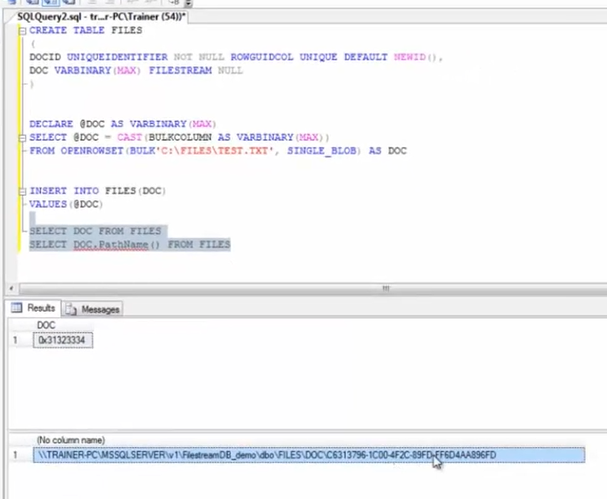
From openrowset (bulk ‘C:\files\test.txt’, single\_blob) as doc

Insert into files(doc)

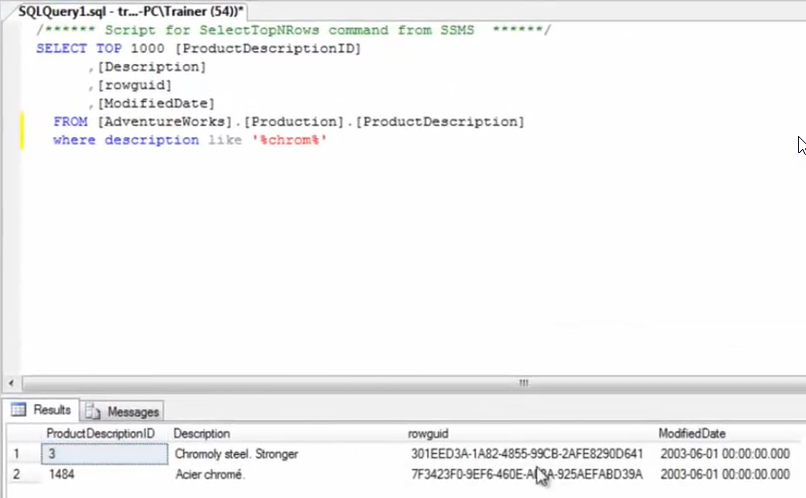
Values(@doc)

Select \* from files

Select doc.Pathname() from files

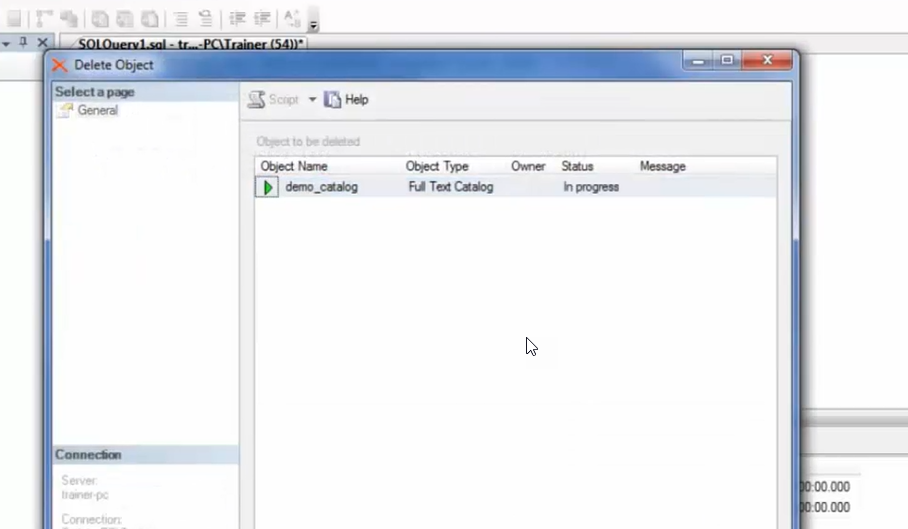


**SQL-free text Search**

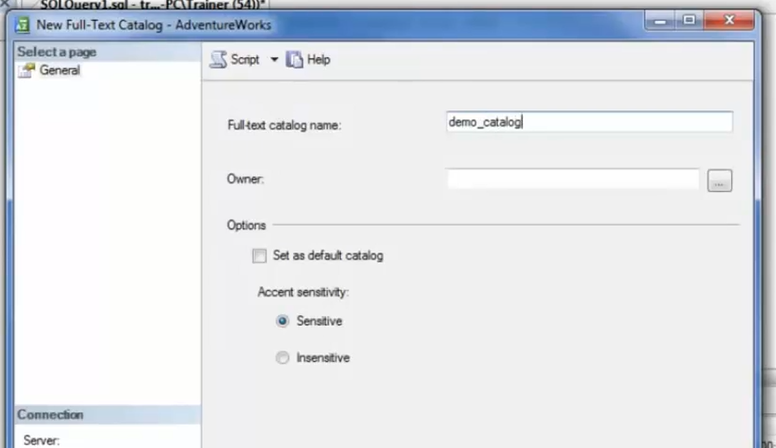


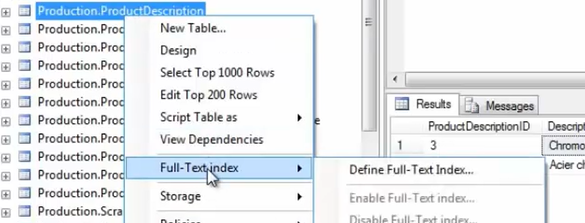
If searching ie , internet explorer

In storage there is a catalog



Right click > create catalogue







Select \* from table

Where freetext(\*, ‘strong and sturdy’)

Select \* from table

Where contains(\*, ‘strong’)

Select \* from table

Where contains(\*, ‘ ”strong” OR “sturdy” ’)

Select \* from table

Where contains(\*, ‘ strong NEAR sturdy ’)

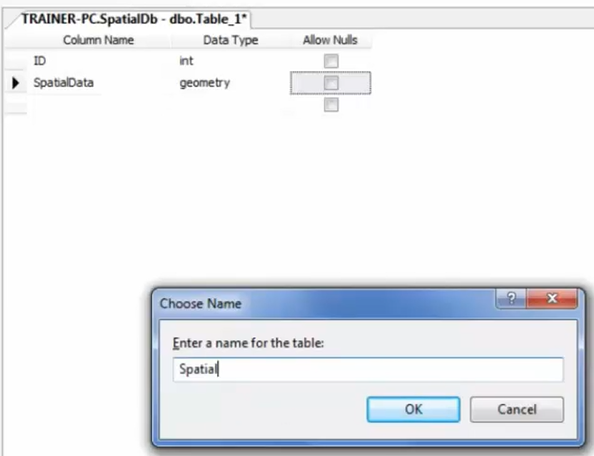
Select \* from table

Where contains(\*, ‘ strong NEAR sturdy ’)

Select \* from table

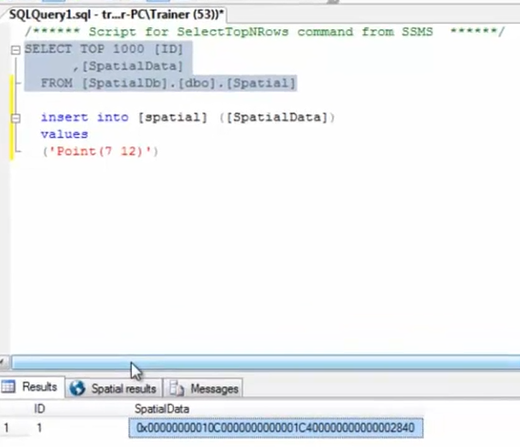
Where contains(\*, ‘ formsof (inflectional, strong) ’)

Geography and Geometry datatypes



Insert into [spatial]([spatialData])

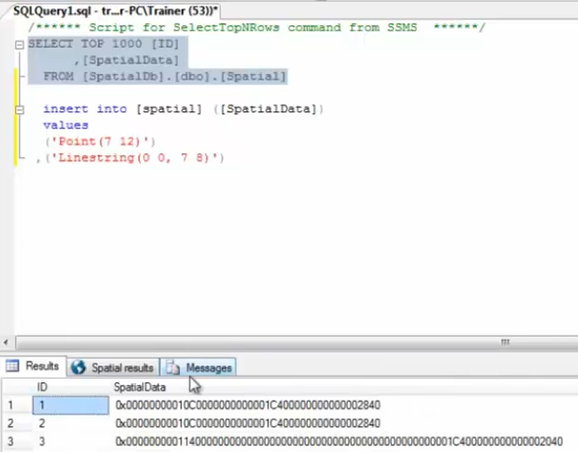
Values (‘Point(7 12)’)

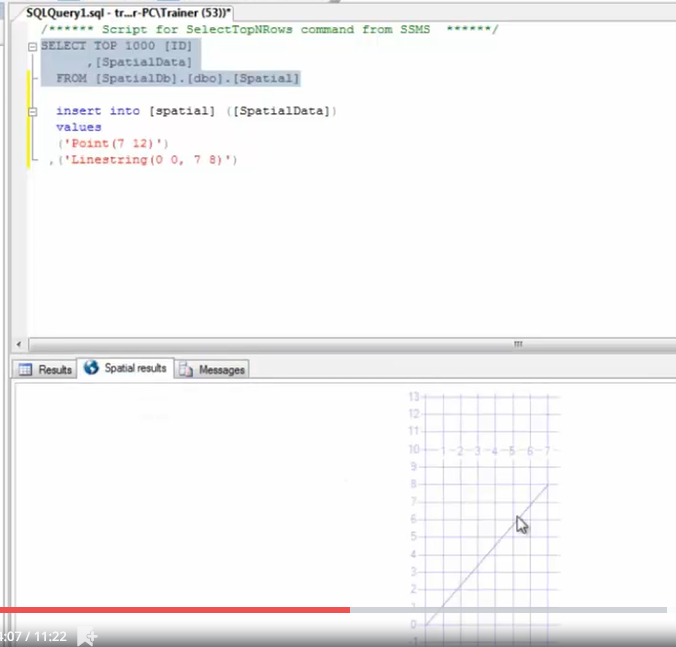


Insert into [spatial] ([spatialData])

Values(‘Point(7,12)’)

,(‘Linestring(0 0, 7 8)’)



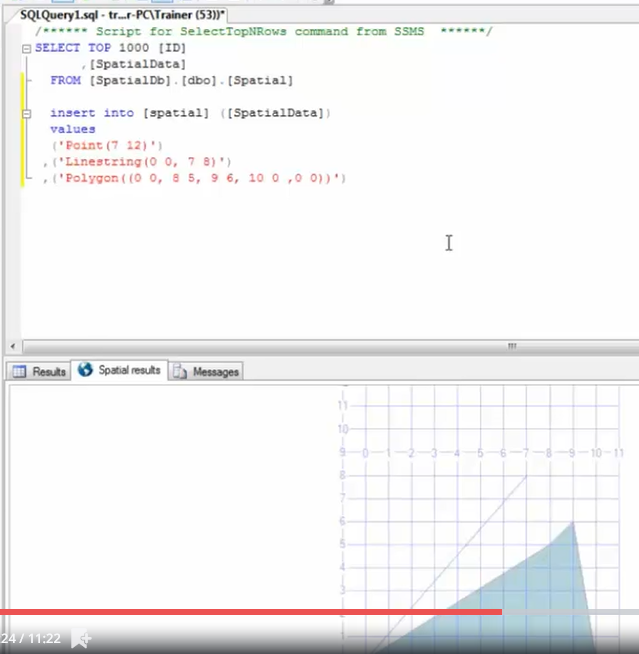


Insert into [spatial] ([spatialData])

Values(‘Point(7,12)’)

,(‘Linestring(0 0, 7 8)’)

,(‘Polygon((0 0, 8 5, 9 6, 10 0, 0 0))’)

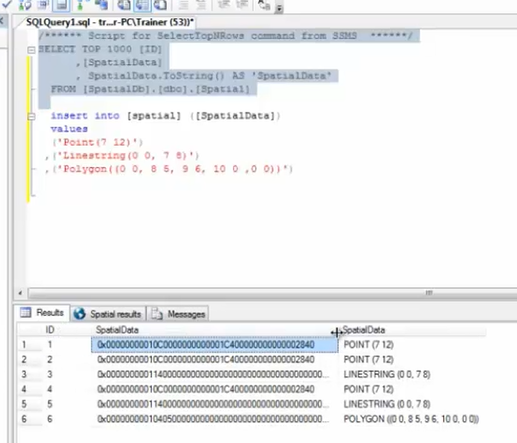


Select top 1000 [ID]

,[spatialData]

,‘spatialData.tostring() as ‘SpecialData’

From [SpatialDb].[dbo].[spatial]



Select top 1000 [ID]

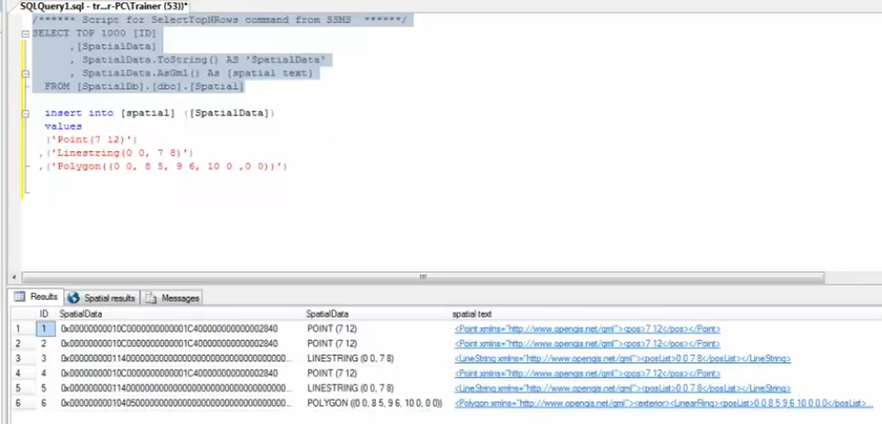
,[spatialData]

,‘spatialData.tostring() as ‘SpecialData’

,SpatialData.AsGml() as [Spatial text]

From [SpatialDb].[dbo].[spatial]

--kind of xml representation



Geography data

Population in each state

Geography info – shape and population info

<http://www.sharepgis.net/page/shape2SQL.aspx>

shape2SQl tool – helps in uploading shapefile and create necessary table for you

<http://vbstech.com/usa-data.aspx>

demo shapefiles

