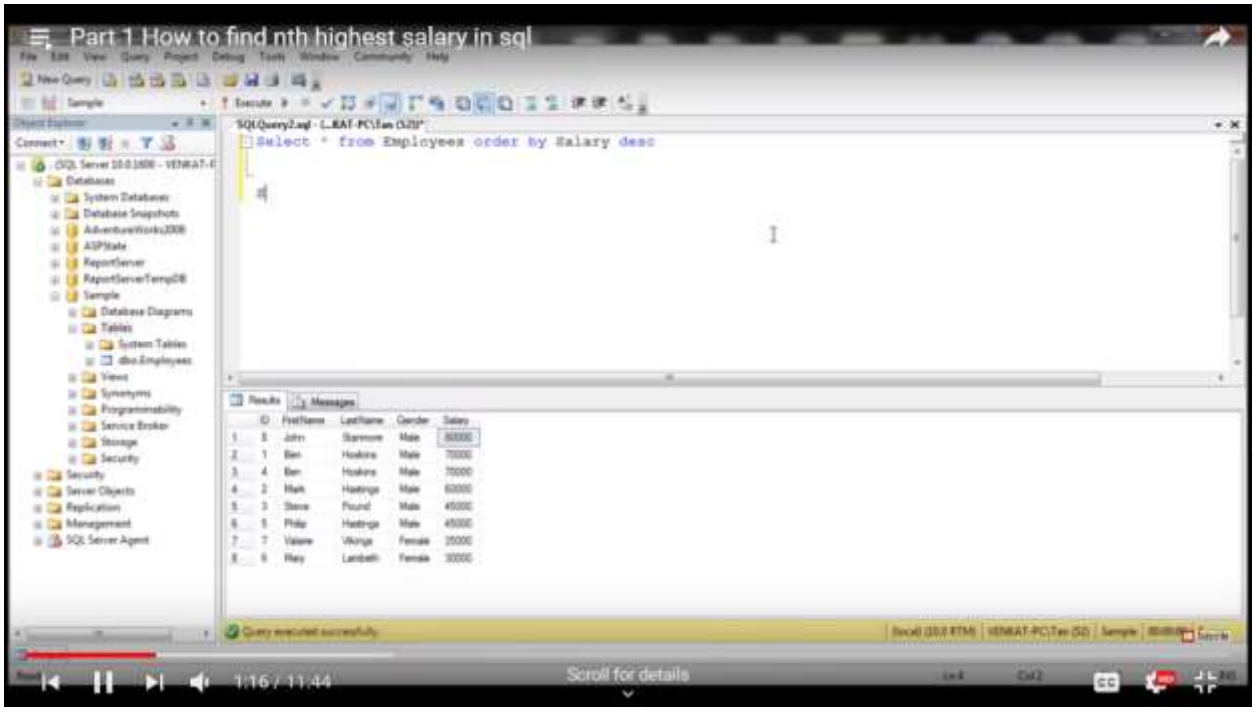
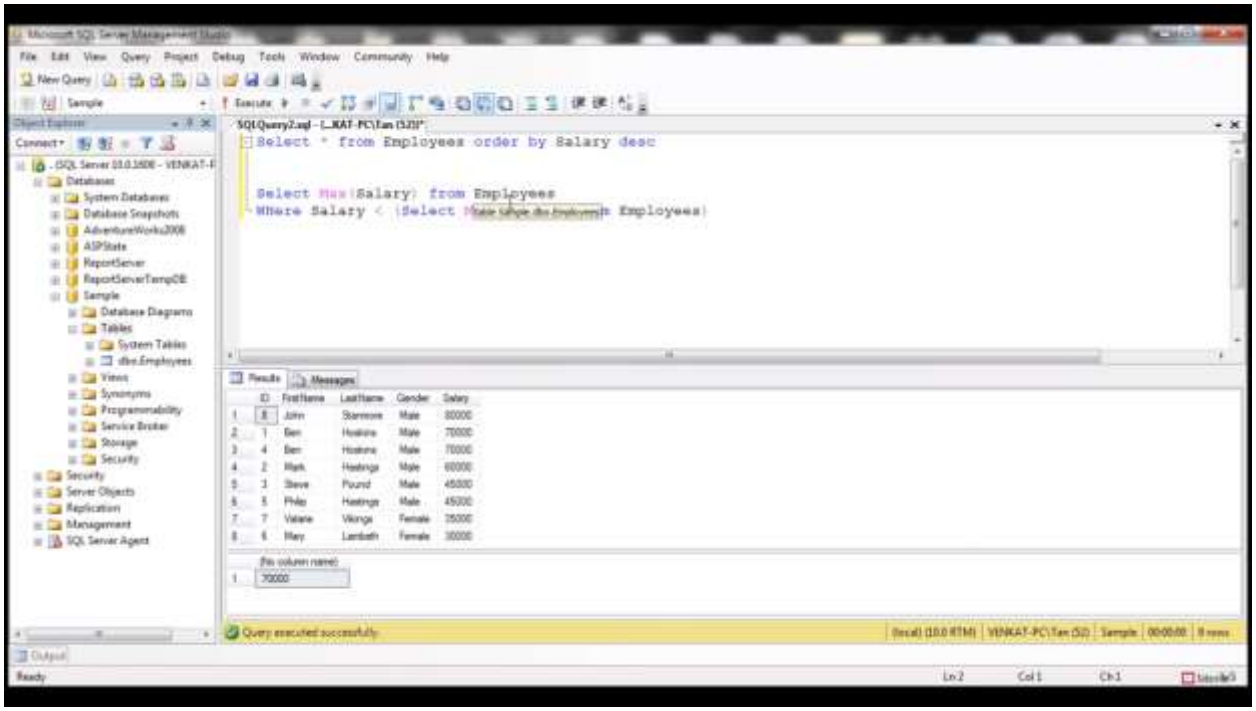
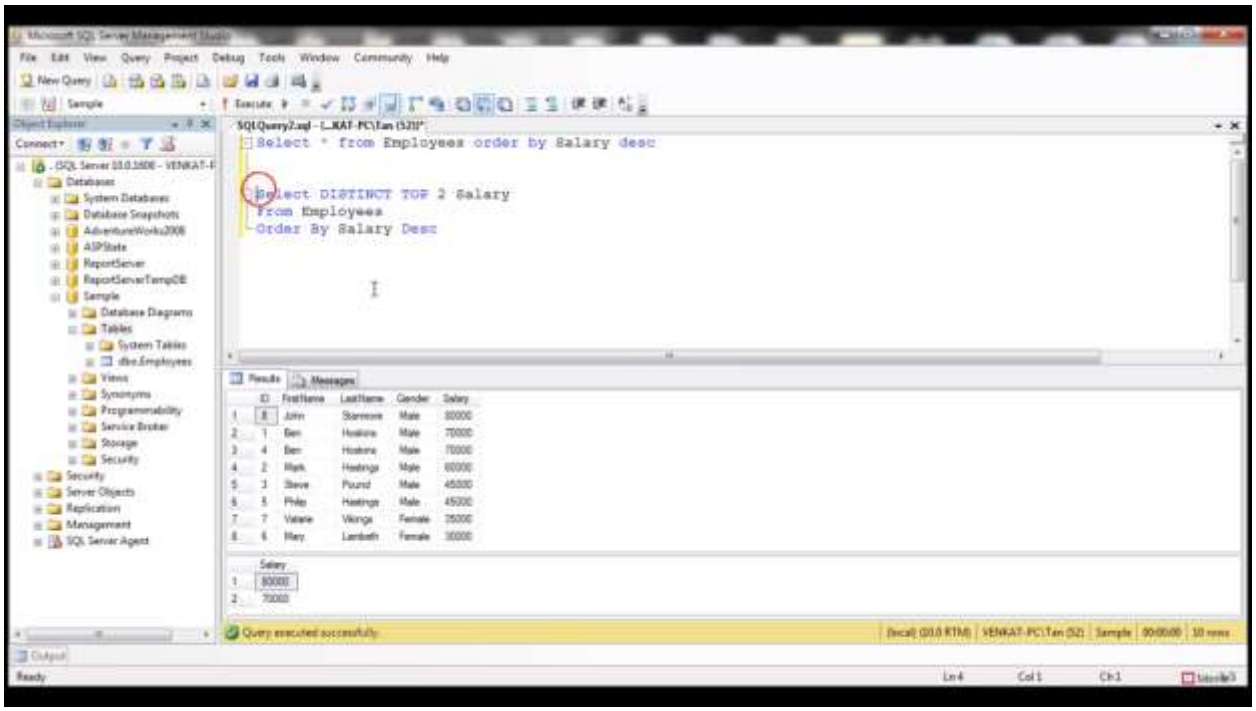
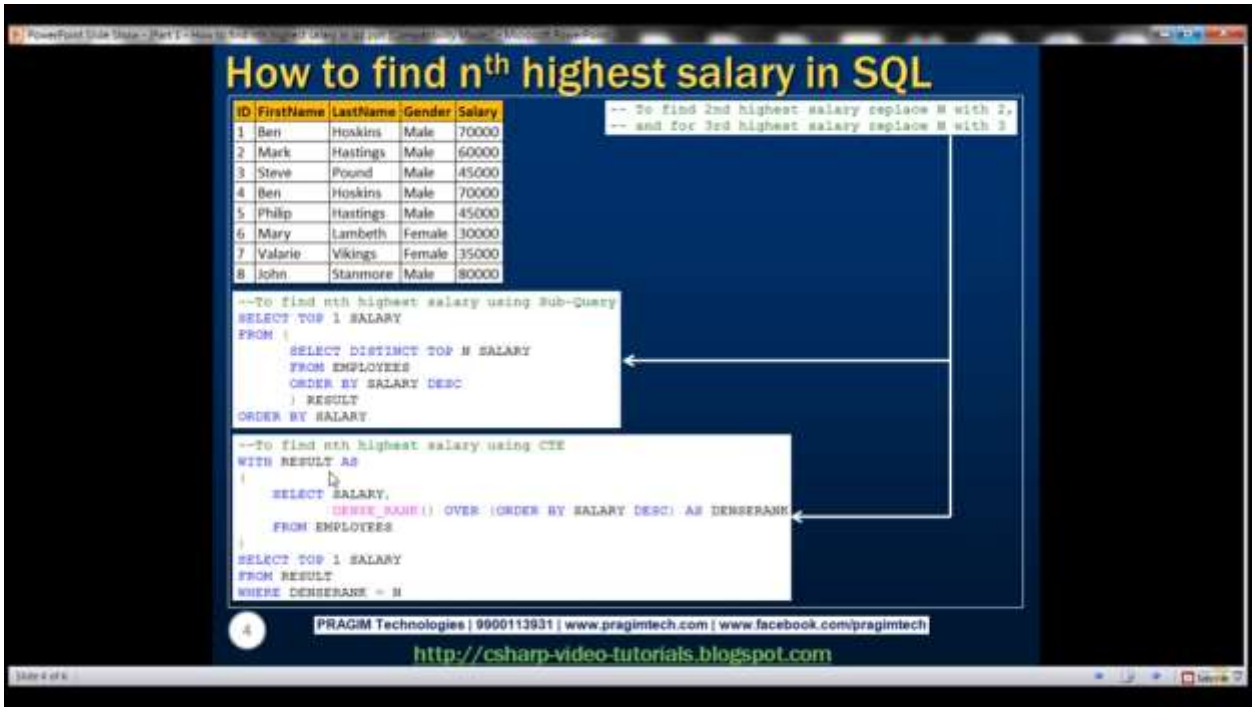
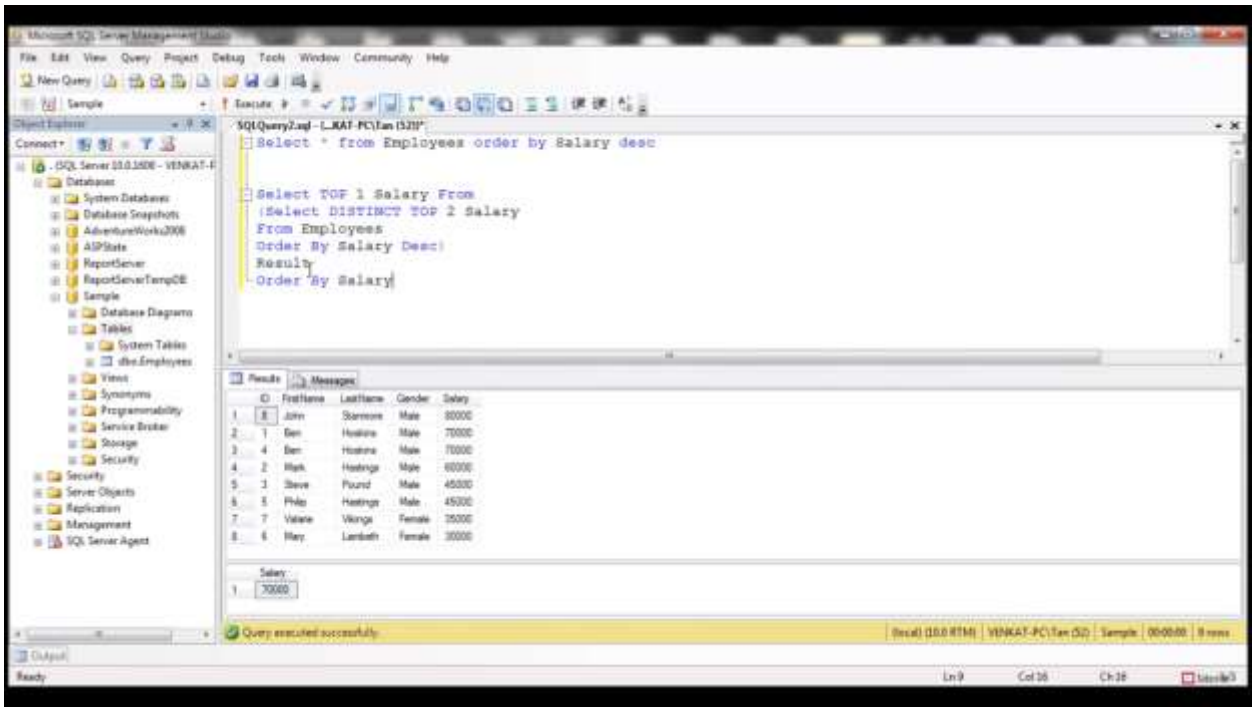
How to find nth highest salary in sql

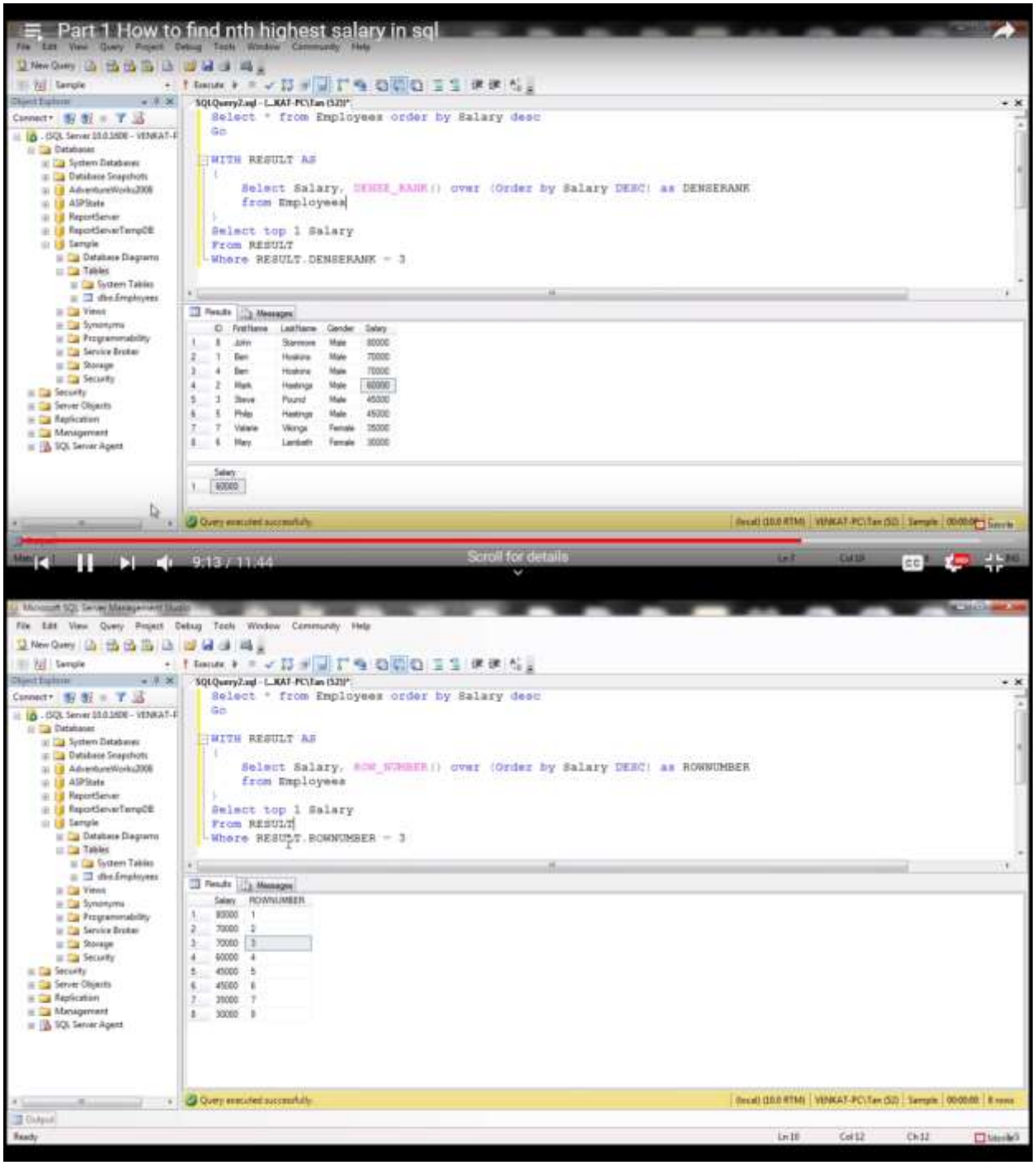



Select * from Employees order by Salary desc
Select Max(Salary) from Employees
Where Salary < (Select Max(Salary) from Employees)
Query executed successfully.

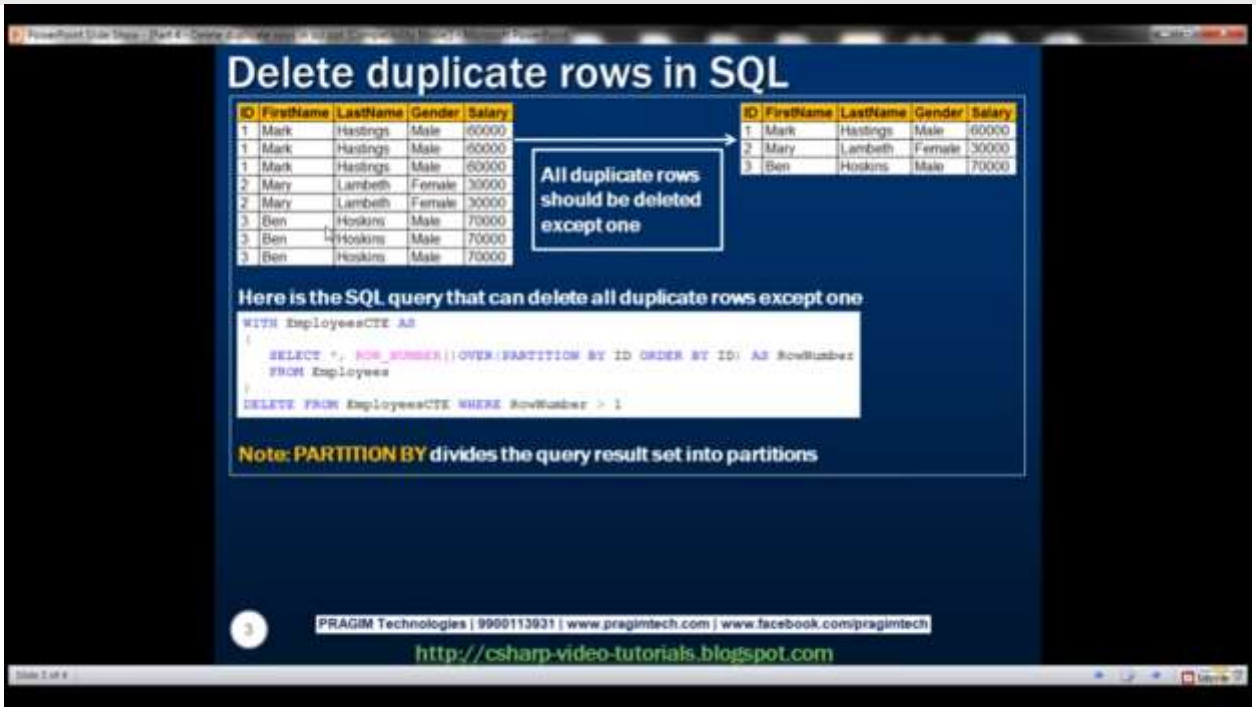

Select * from Employees order by Salary desc
Select DISTINCT TOP 2 Salary
From Employees
Order By Salary Desc
Query executed successfully.

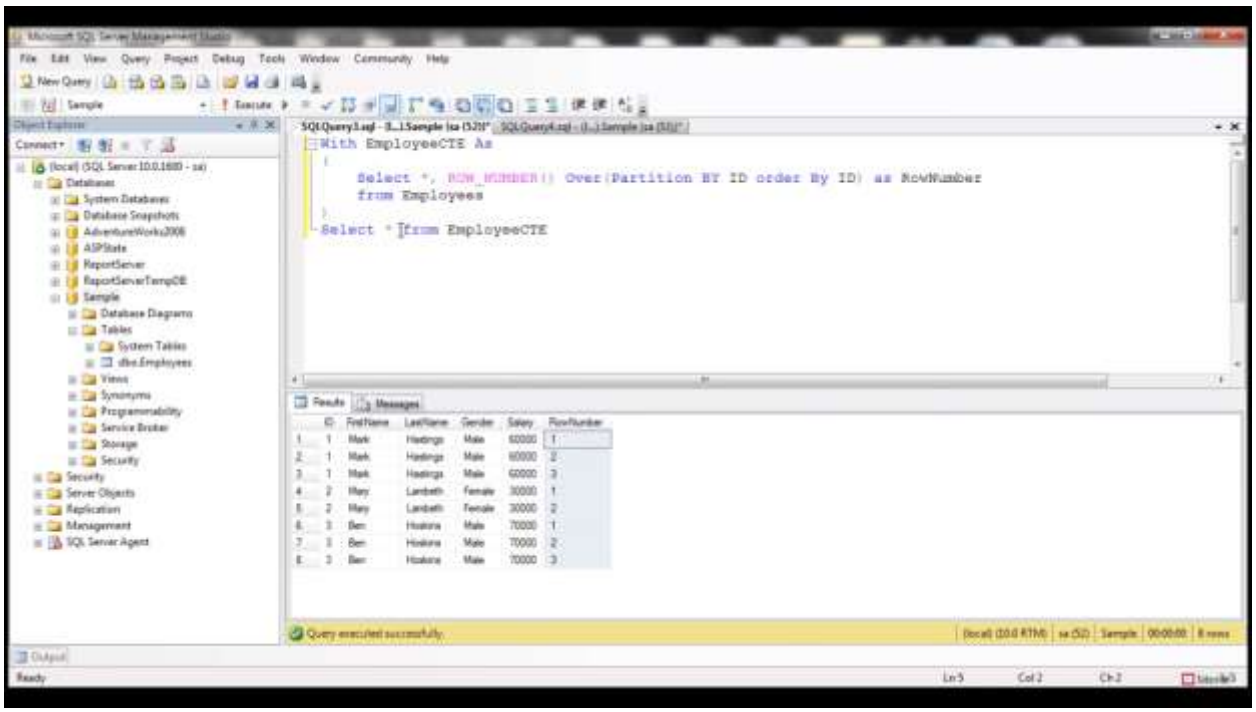

How to find nth highest salary in SQL
ID FirstName LastName Gender Salary
1 Ben Hoskins Male 70000
2 Mark Hastings Male 60000
3 Steve Pound Male 45000
4 Ben Hoskins Male 70000
5 Philp Hastings Male 45000
6 Mary Lambeth Female 30000
7 Valarie Vikings Female 35000
8 John Stanmore Male 80000
-- To find 2nd highest salary replace N with 2,
-- and for 3rd highest salary replace N with 3
--To find nth highest salary using Sub-Query
SELECT TOP 1 SALARY
FROM (
SELECT DISTINCT TOP N SALARY
FROM EMPLOYEES
ORDER BY SALARY DESC
) RESULT
ORDER BY SALARY
--To find nth highest salary using CTE
WITH RESULT AS
(
SELECT SALARY,
DENSE_RANK() OVER (ORDER BY SALARY DESC) AS DENSERANK
FROM EMPLOYEES
)
SELECT TOP 1 SALARY
FROM RESULT
WHERE DENSERANK = N
4
PRAGIM Technologies | 9900113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com

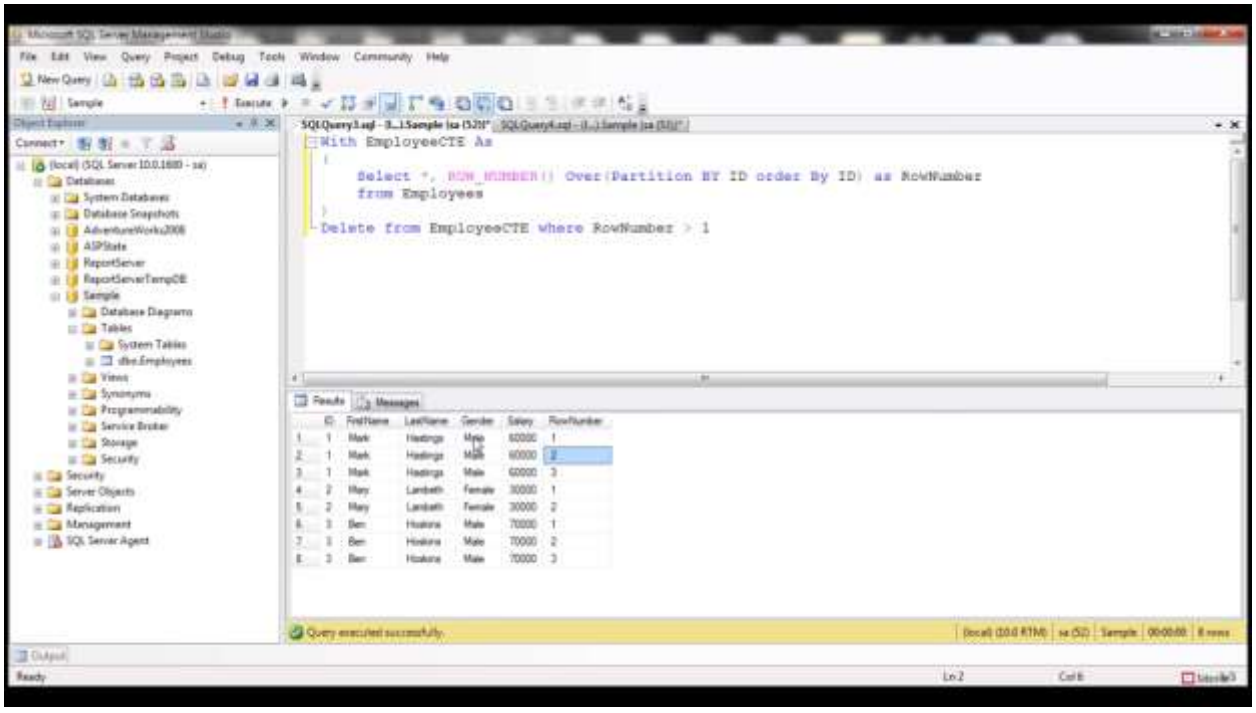

Select * from Employees order by Salary desc
Select TOP 1 Salary From
(Select DISTINCT TOP 2 Salary
From Employees
Order By Salary Desc)
Result
Order By Salary

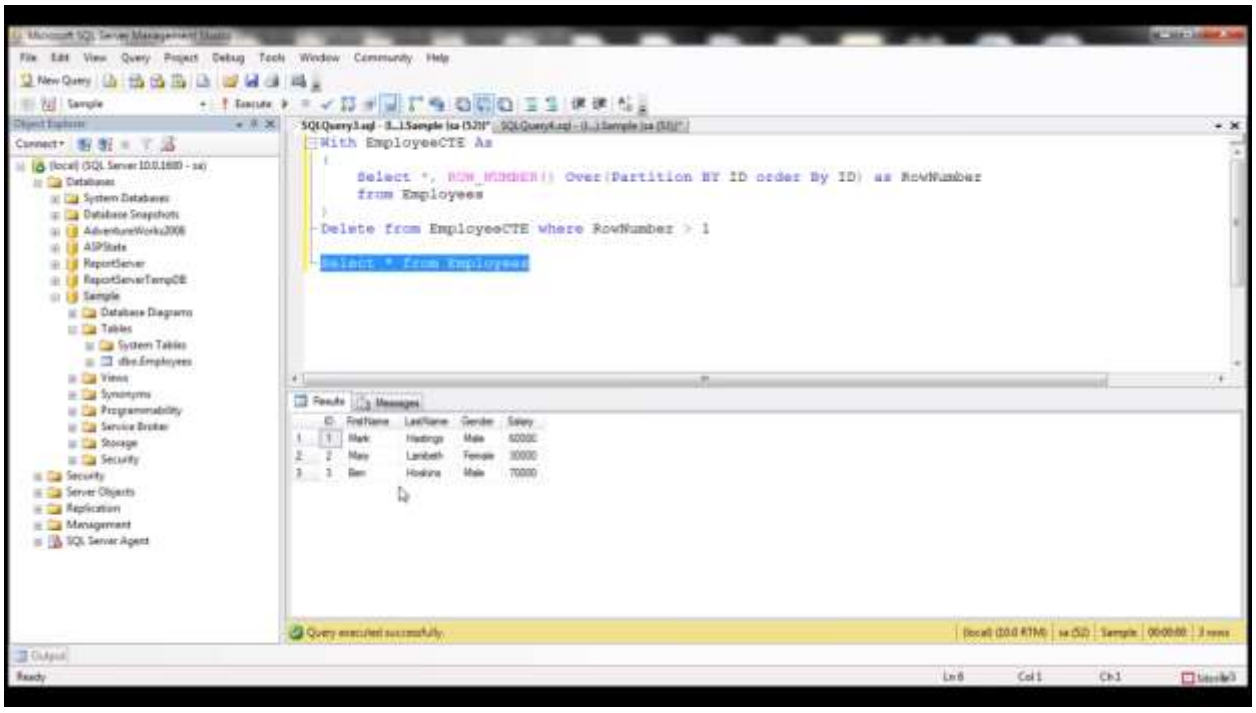


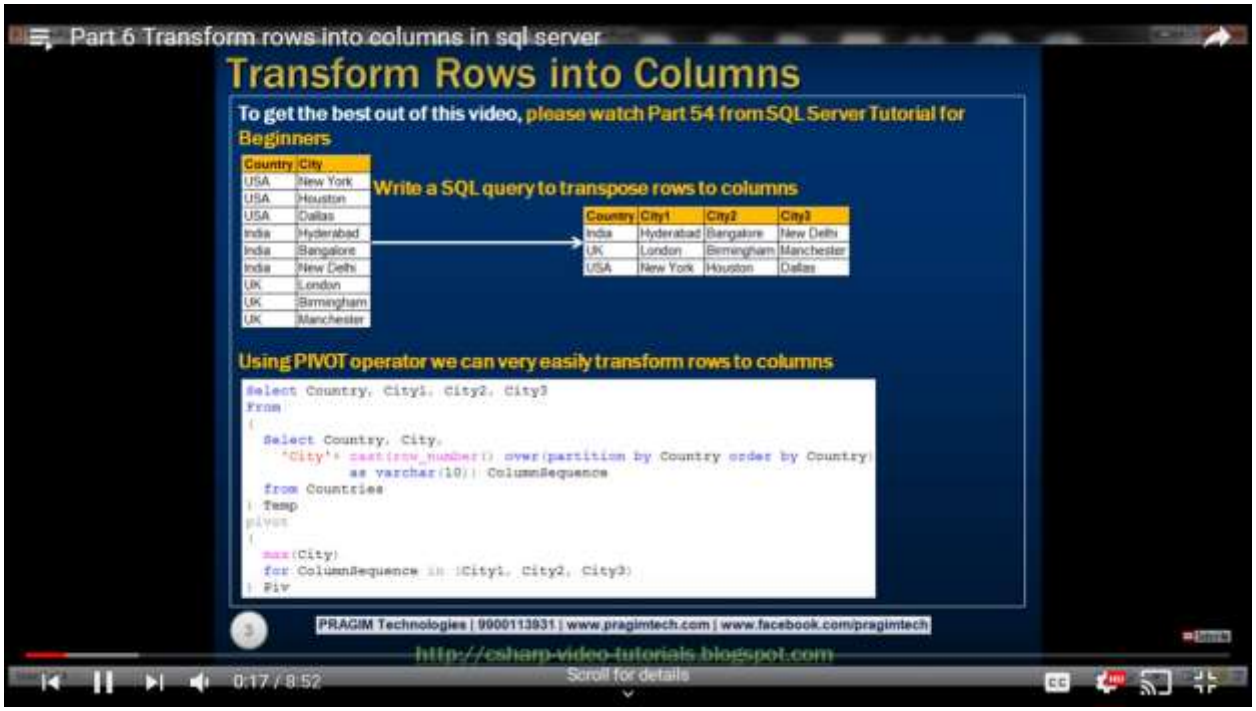

```
SELECT * /*This is the outer query part */
FROM Employee Emp1
WHERE (N-1) = ( /* Subquery starts here */
SELECT COUNT(DISTINCT(Emp2.Salary))
FROM Employee Emp2 WHERE Emp2.Salary > Emp1.Salary)
```

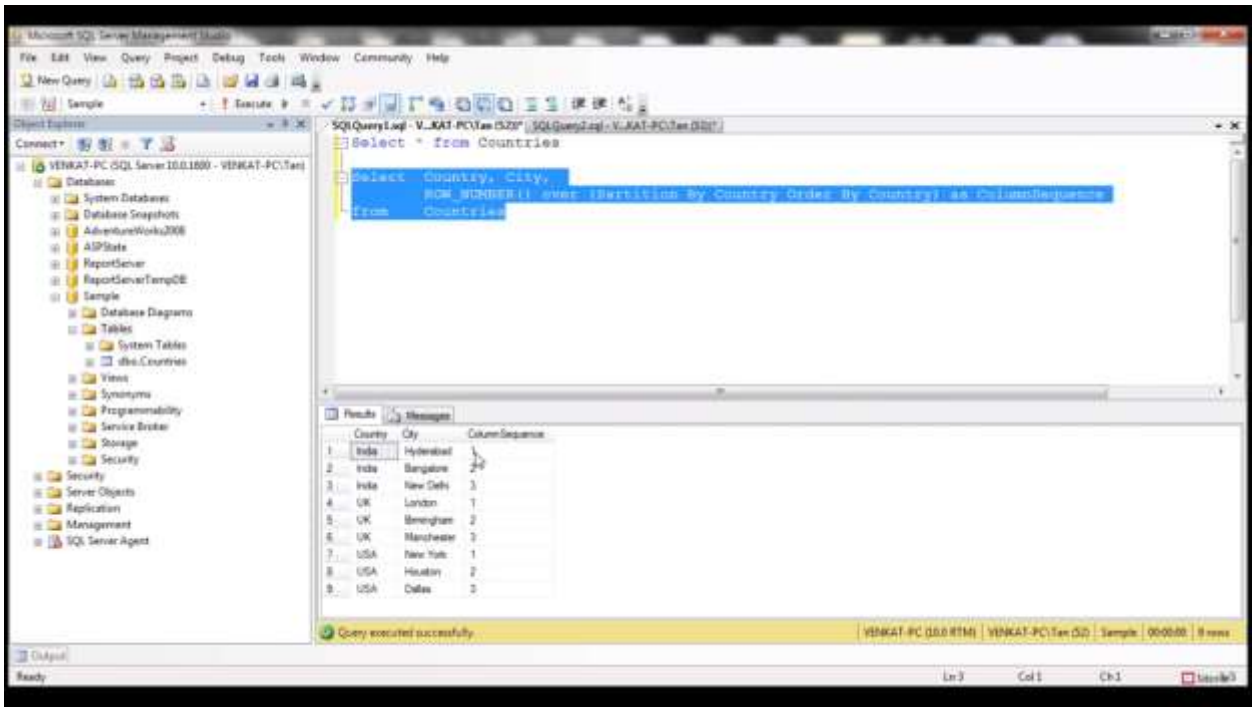

Delete duplicate rows in SQL
ID FirstName LastName Gender Salary
1 Mark Hastings Male 60000
1 Mark Hastings Male 60000
1 Mark Hastings Male 60000
2 Mary Lambeth Female 30000
2 Mary Lambeth Female 30000
3 Ben Hoskins Male 70000
3 Ben Hoskins Male 70000
3 Ben Hoskins Male 70000
All duplicate rows should be deleted except one
ID FirstName LastName Gender Salary
1 Mark Hastings Male 60000
2 Mary Lambeth Female 30000
3 Ben Hoskins Male 70000
Here is the SQL query that can delete all duplicate rows except one
WITH EmployeesCTE AS
(
SELECT *, ROW_NUMBER()OVER(PARTITION BY ID ORDER BY ID) AS RowNumber
FROM Employees
)
DELETE FROM EmployeesCTE WHERE RowNumber > 1
Note: PARTITION BY divides the query result set into partitions
3
PRAGIM Technologies | 9900113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com

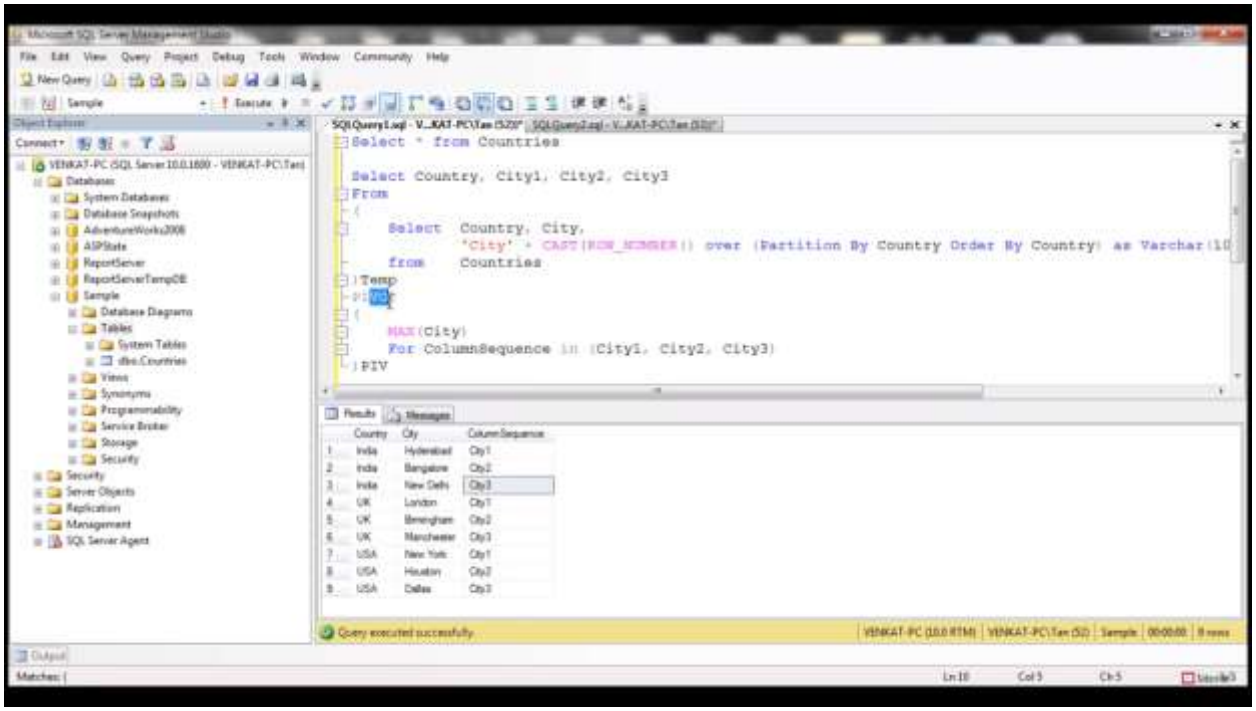

With EmployeeCTE As
(
Select *, ROW_NUMBER() Over(Partition BY ID order By ID) as RowNumber
from Employees
)
Select * from EmployeeCTE

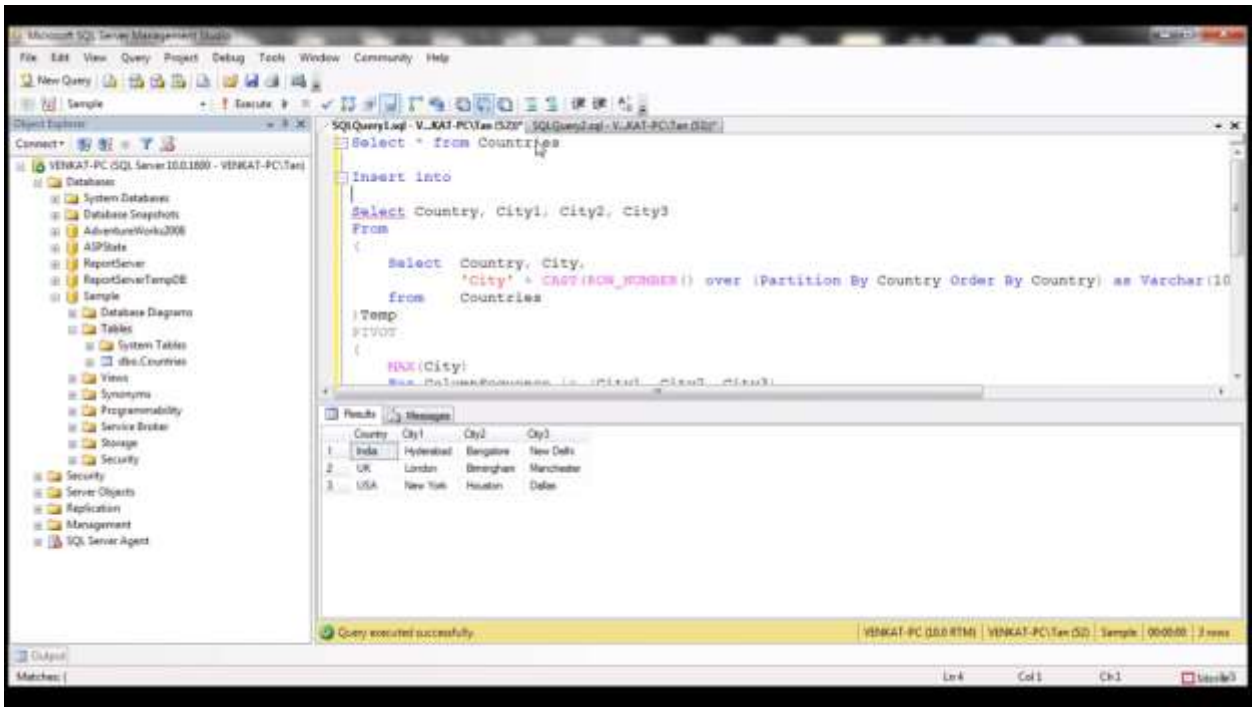

With EmployeeCTE As
(
Select *, ROW_NUMBER() Over(Partition BY ID order By ID) as RowNumber
from Employees
)
Delete from EmployeeCTE where RowNumber > 1

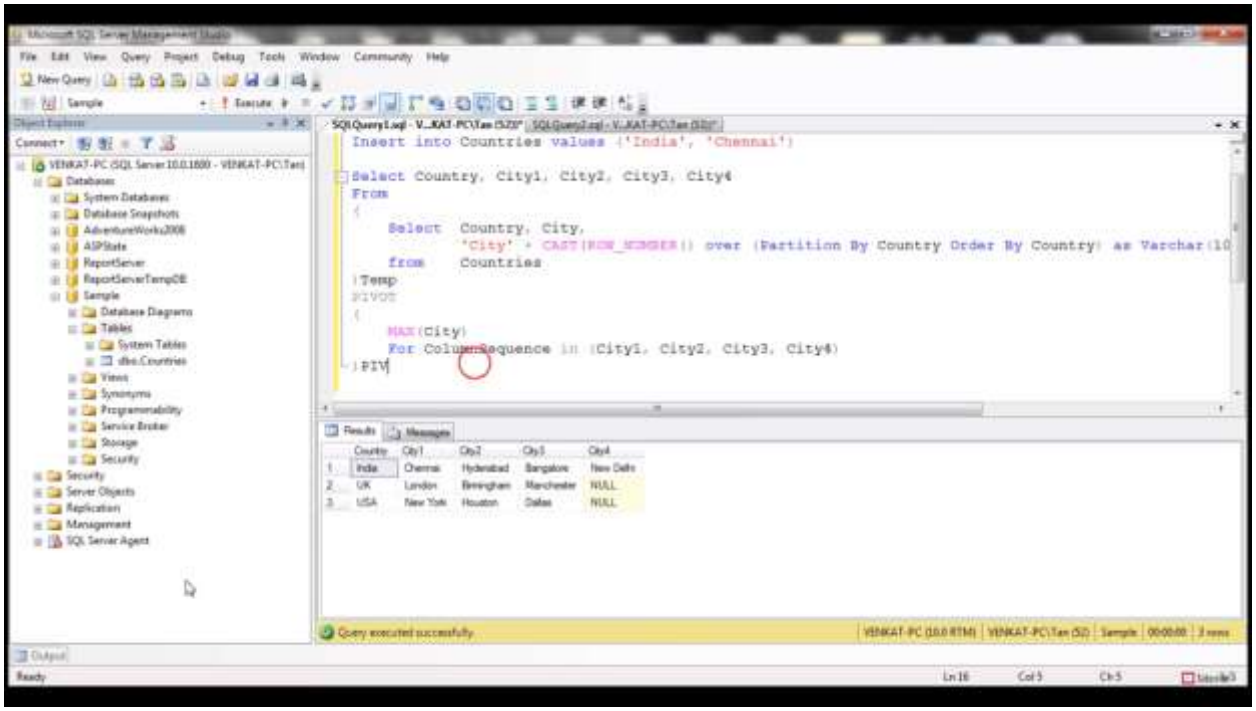

With EmployeeCTE As
(
Select *, ROW_NUMBER() Over(Partition BY ID order By ID) as RowNumber
from Employees
)
Delete from EmployeeCTE where RowNumber > 1
Select * from Employees

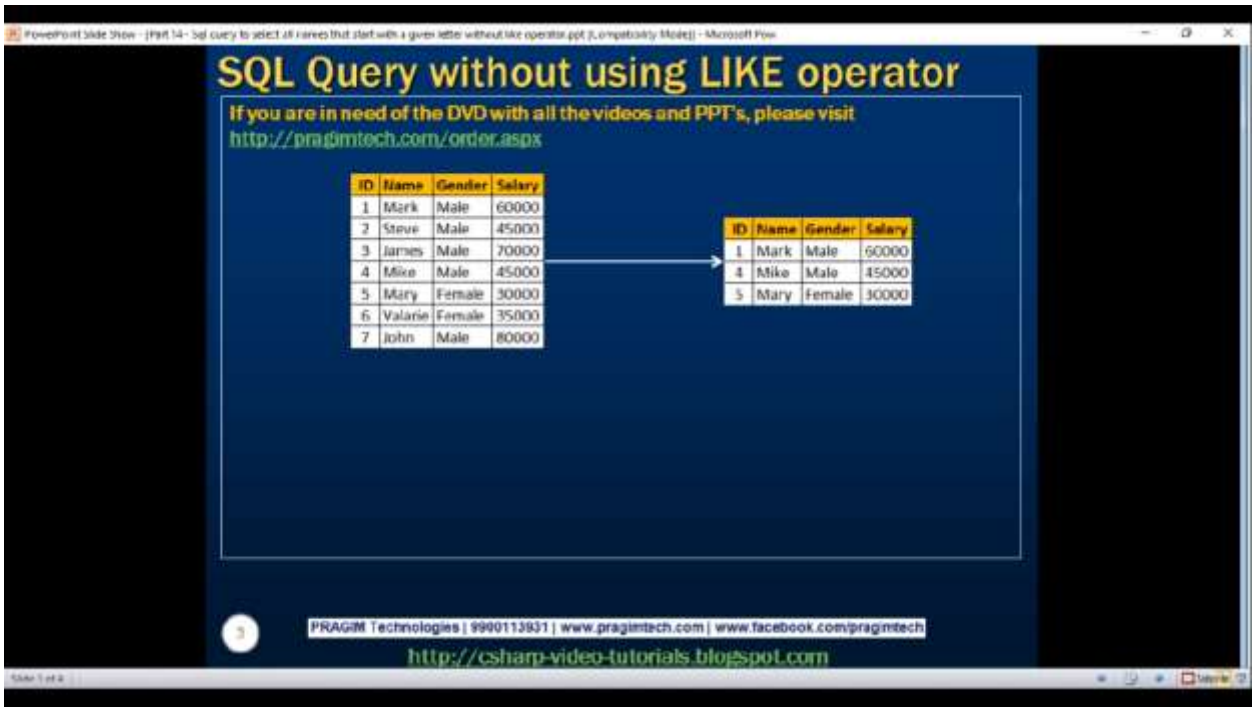

Part 6 Transform rows into columns in sql server
Transform Rows into Columns
To get the best out of this video, please watch Part 54 from SQL Server Tutorial for Beginners
Write a SQL query to transpose rows to columns
Using PIVOT operator we can very easily transform rows to columns
PRAGIM Technologies | 9000113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com
0:17 / 8:52
Scroll for details

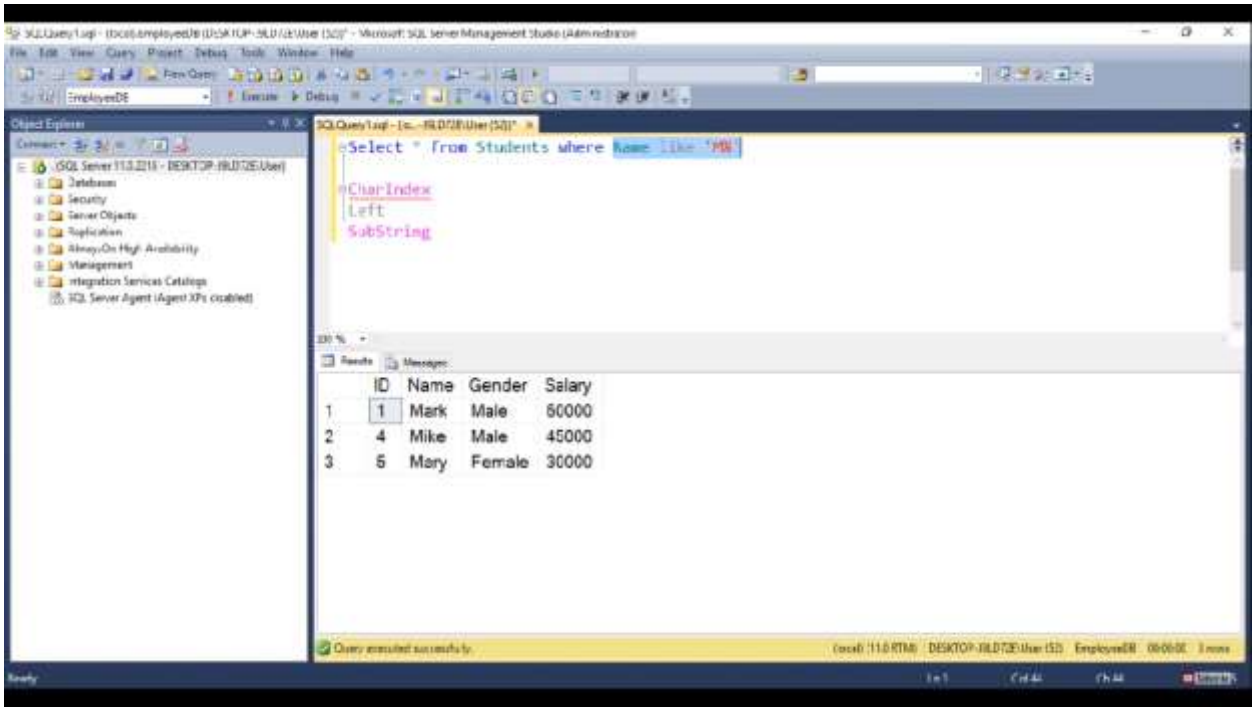

Select * from Countries
Select Country, City,
ROW_NUMBER() over (Partition By Country Order By Country) as ColumnSequence
from Countries
Query executed successfully.

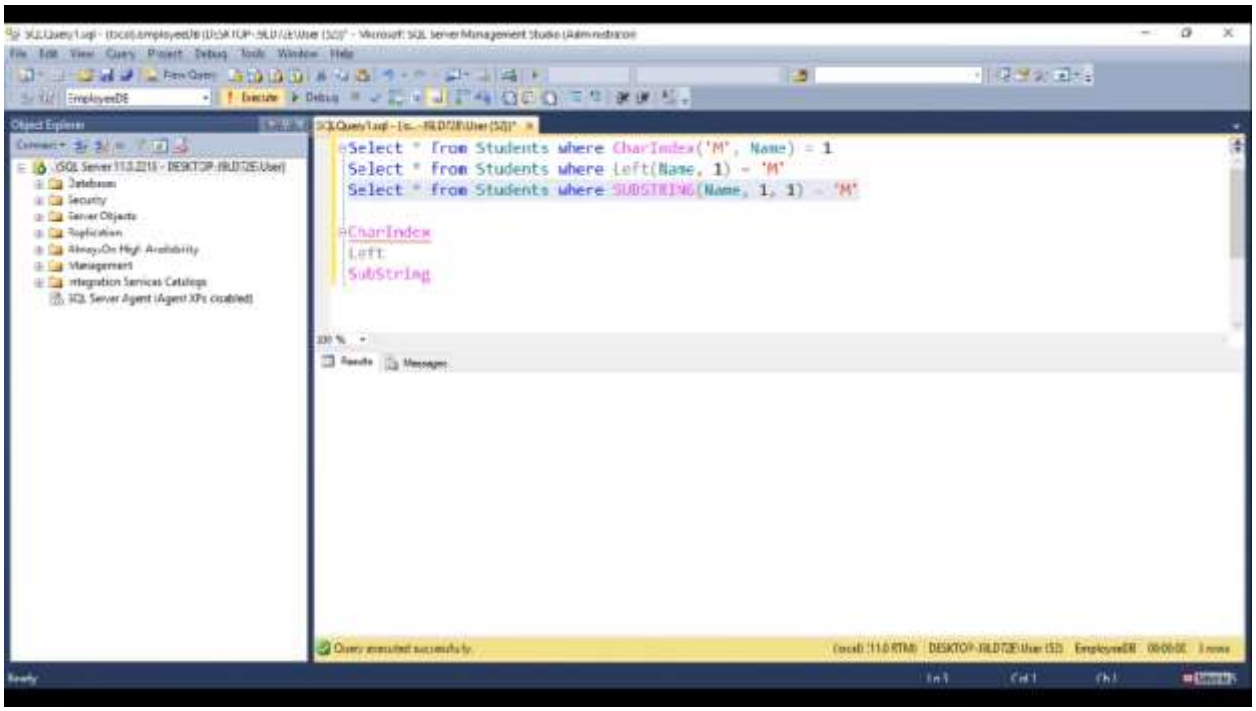

Select * from Countries
Select Country, City1, City2, City3
From
(
Select Country, City,
'City' + CAST(ROW_NUMBER() over (Partition By Country Order By Country) as Varchar(10
from Countries
)Temp
PIVOT
(
MAX(City)
For ColumnSequence in (City1, City2, City3)
)PIV
Country City ColumnSequence
1 India Hyderabad City1
2 India Bangalore City2
3 India New Delhi City3
4 UK London City1
5 UK Birmingham City2
6 UK Manchester City3
7 USA New York City1
8 USA Houston City2
9 USA Dallas City3
Query executed successfully.

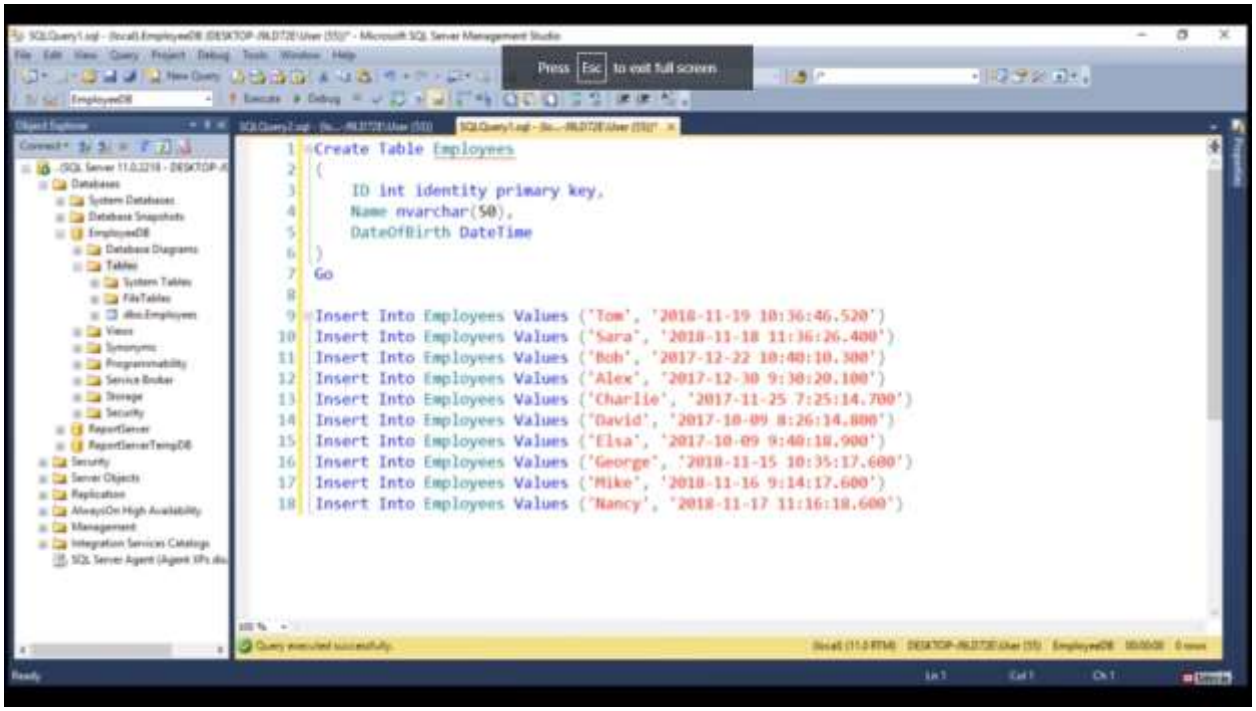

Select * from Countries
Insert into
Select Country, City1, City2, City3
From
(
Select Country, City,
'City' + CAST(ROW_NUMBER() over (Partition By Country Order By Country) as Varchar(10
from Countries
)Temp
PIVOT
(
MAX(City)
Country City1 City2 City3
1 India Hyderabad Bangalore New Delhi
2 UK London Birmingham Manchester
3 USA New York Houston Dallas
Query executed successfully.


Insert into Countries values ('India', 'Chennai')
Select Country, City1, City2, City3, City4
From
(
Select Country, City,
'City' + CAST(ROW_NUMBER() over (Partition By Country Order By Country) as Varchar(10
from Countries
)Temp
PIVOT
(
MAX(City)
For ColumnSequence in (City1, City2, City3, City4)
)PIV
Country City1 City2 City3 City4
1 India Chennai Hyderabad Bangalore New Delhi
2 UK London Birmingham Manchester NULL
3 USA New York Houston Dallas NULL
Query executed successfully.

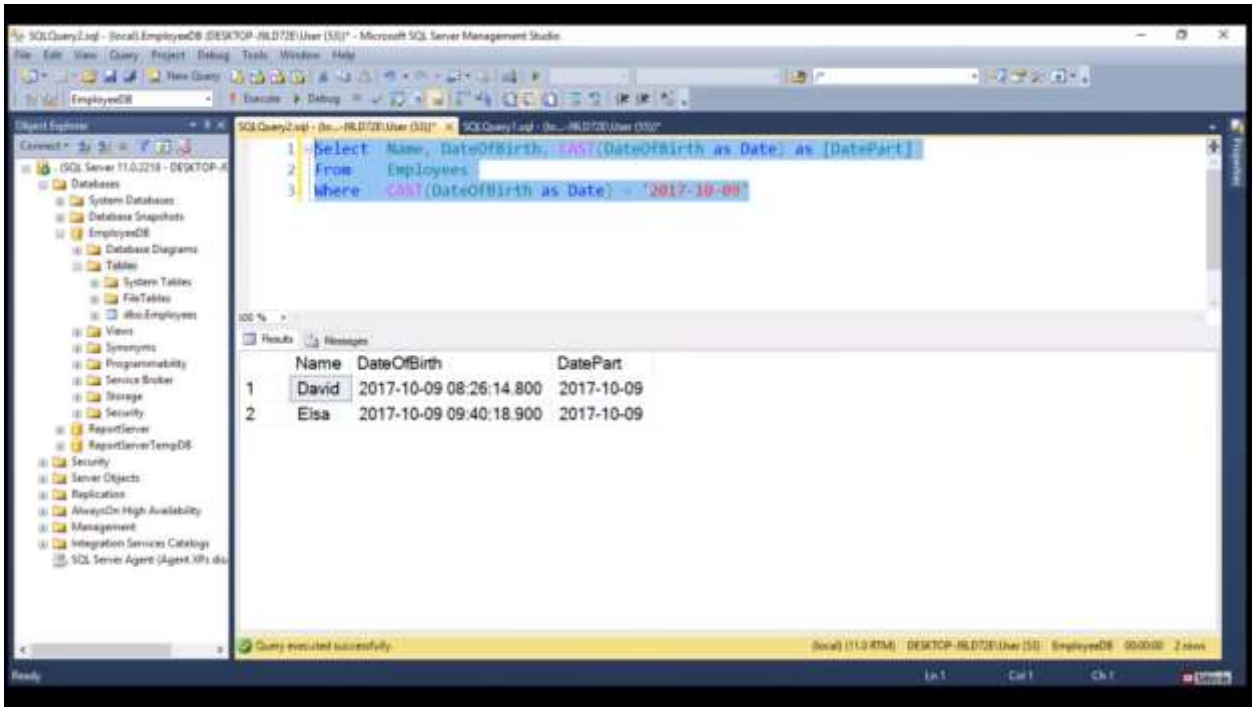

SQL Query without using LIKE operator
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
ID Name Gender Salary
1 Mark Male 60000
2 Steve Male 45000
3 James Male 70000
4 Mike Male 45000
5 Mary Female 30000
6 Valarie Female 35000
7 John Male 80000
ID Name Gender Salary
1 Mark Male 60000
4 Mike Male 45000
5 Mary Female 30000
PRAGIM Technologies | 9990113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com

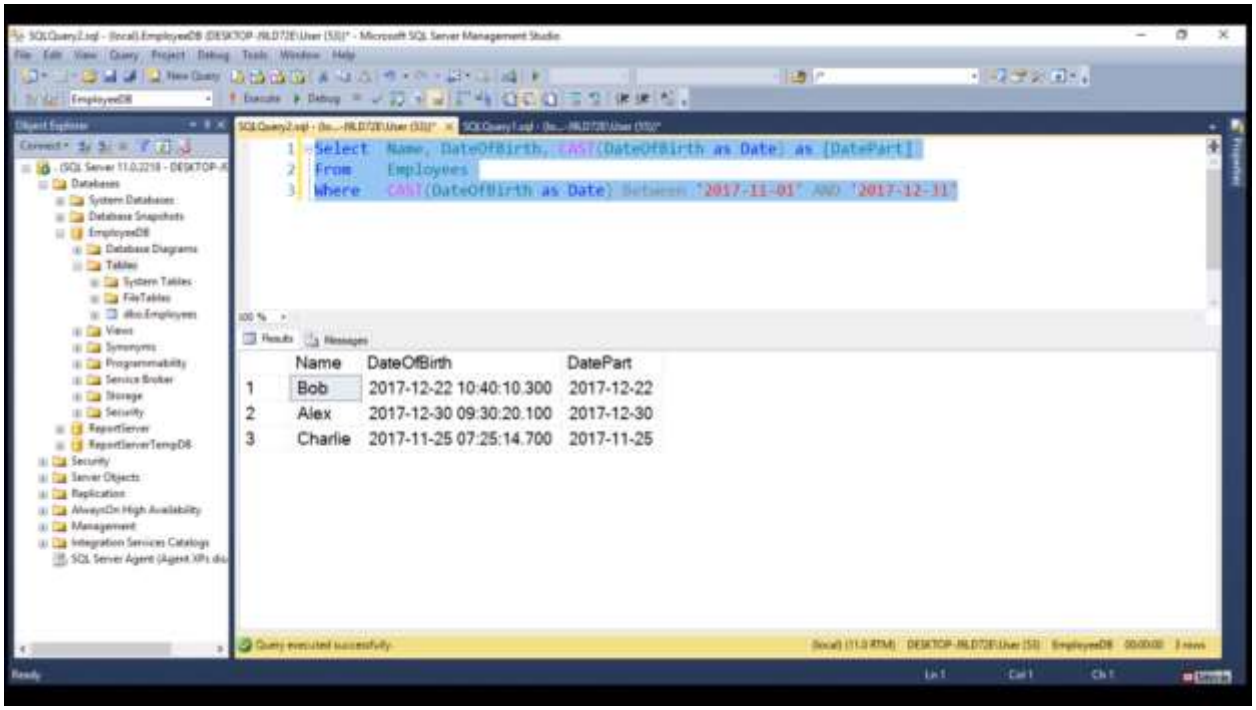

Select * from Students where Name like 'M%'
CharIndex
Left
SubString
ID Name Gender Salary
1 1 Mark Male 60000
2 4 Mike Male 45000
3 5 Mary Female 30000

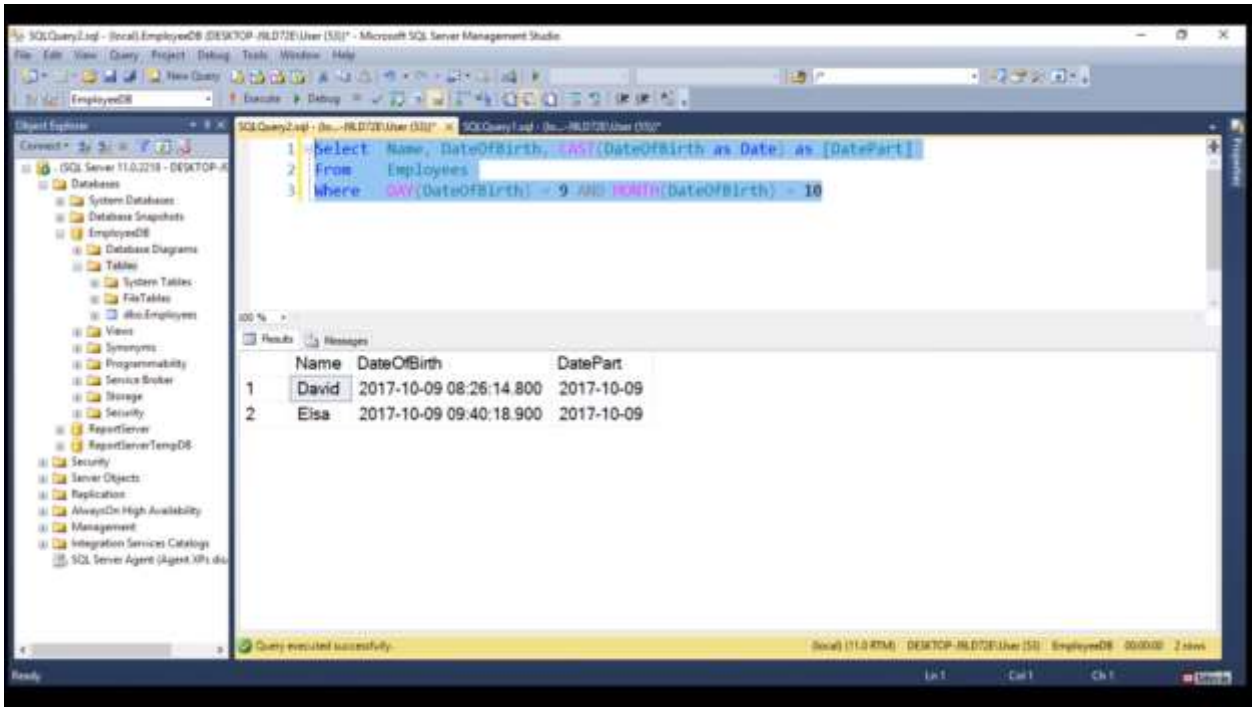

Select * from Students where CharIndex('M', Name) = 1
Select * from Students where Left(Name, 1) = 'M'
Select * from Students where SUBSTRING(Name, 1, 1) = 'M'
CharIndex
Left
SubString


Press Esc to exit full screen
Create Table Employees
(
ID int identity primary key,
Name nvarchar(50),
DateOfBirth DateTime
)
Go
Insert Into Employees Values ('Tom', '2018-11-19 10:36:46.520')
Insert Into Employees Values ('Sara', '2018-11-18 11:36:26.400')
Insert Into Employees Values ('Bob', '2017-12-22 10:40:10.300')
Insert Into Employees Values ('Alex', '2017-12-30 9:30:20.100')
Insert Into Employees Values ('Charlie', '2017-11-25 7:25:14.700')
Insert Into Employees Values ('David', '2017-10-09 8:26:14.800')
Insert Into Employees Values ('Elsa', '2017-10-09 9:40:18.900')
Insert Into Employees Values ('George', '2018-11-15 10:35:17.600')
Insert Into Employees Values ('Mike', '2018-11-16 9:14:17.600')
Insert Into Employees Values ('Nancy', '2018-11-17 11:16:18.600')

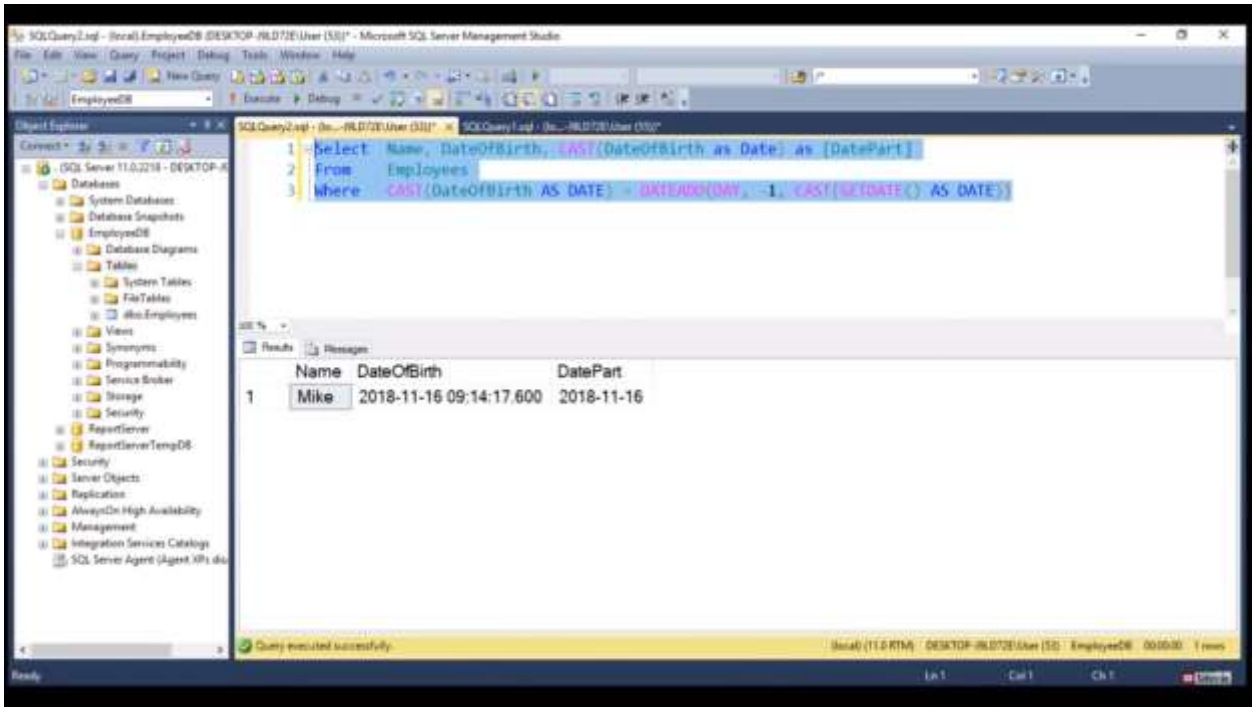

SQL Date Interview Questions
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
ID Name DateOfBirth
1 Tom 2018-11-19 10:36:46.520
2 Sara 2018-11-18 11:36:26.400
3 Bob 2017-12-22 10:40:10.300
4 Alex 2017-12-30 09:30:20.100
5 Charlie 2017-11-25 07:25:14.700
6 David 2017-10-09 08:26:14.800
7 Elsa 2017-10-09 09:40:18.900
8 George 2018-11-15 10:35:17.600
9 Mike 2018-11-16 09:14:17.600
10 Nancy 2018-11-17 11:16:18.600
Write a SQL query to get all
People born on a given date (9th October 2017)
People born between 2 given dates (Nov 1, 2017 & Dec 31, 2017)
People born on the same day and month excluding the year (9th October)
People whose birth year is the same (2017, 2018 etc.)
People born yesterday, today, tomorrow, last seven days, next 7 days etc...
Orders placed between 2 given dates or on a given day, month, year etc...
Customers gained or lost
Employees joined or left etc...
3
PRAGIM Technologies | 9900113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com

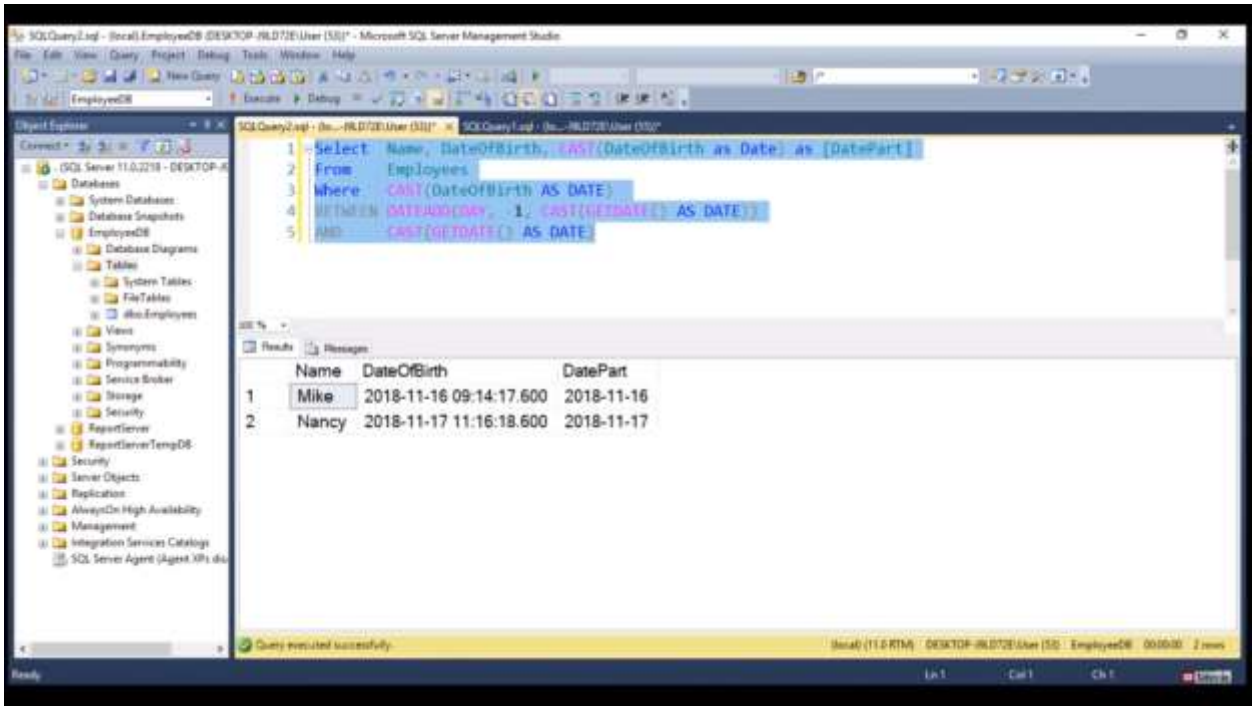

Select Name, DateOfBirth, CAST(DateOfBirth as Date) as [DatePart]
From Employees
Where CAST(DateOfBirth as Date) = '2017-10-09'
Name DateOfBirth DatePart
1 David 2017-10-09 08:26:14.800 2017-10-09
2 Elsa 2017-10-09 09:40:18.900 2017-10-09

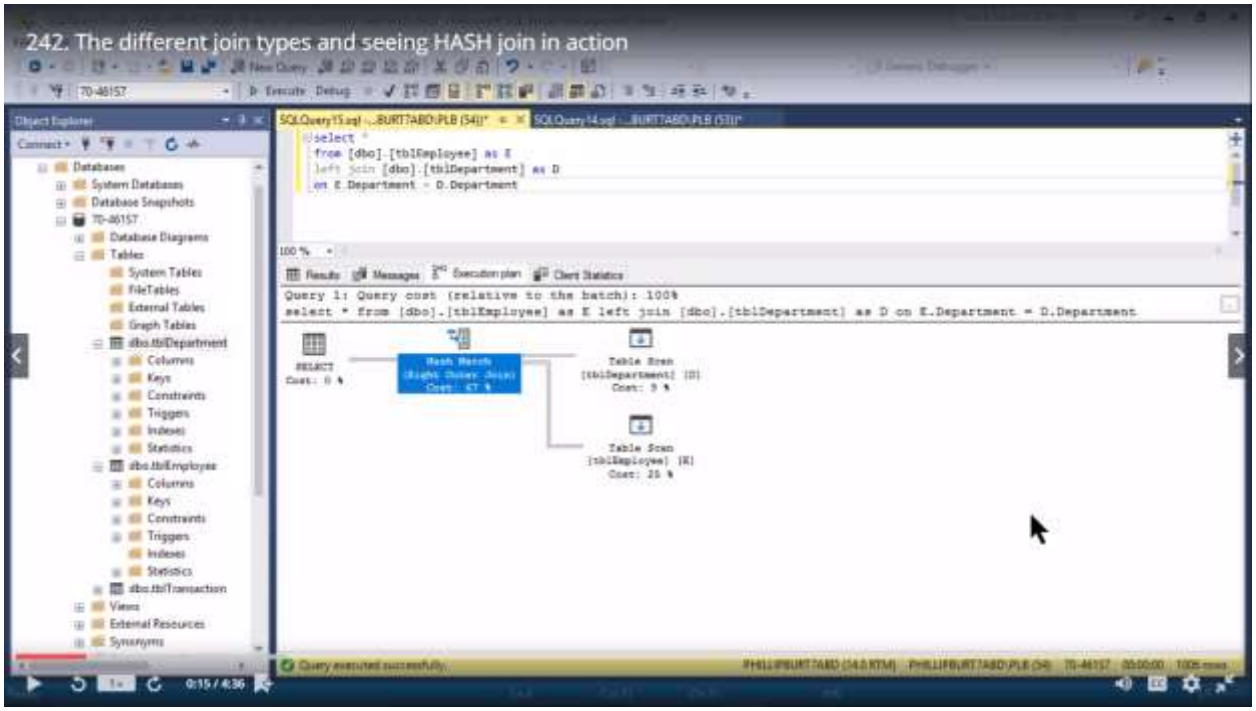

Select Name, DateOfBirth, CAST(DateOfBirth as Date) as [DatePart]
From Employees
Where CAST(DateOfBirth as Date) Between '2017-11-01' AND '2017-12-31'
Name DateOfBirth DatePart
1 Bob 2017-12-22 10:40:10.300 2017-12-22
2 Alex 2017-12-30 09:30:20.100 2017-12-30
3 Charlie 2017-11-25 07:25:14.700 2017-11-25

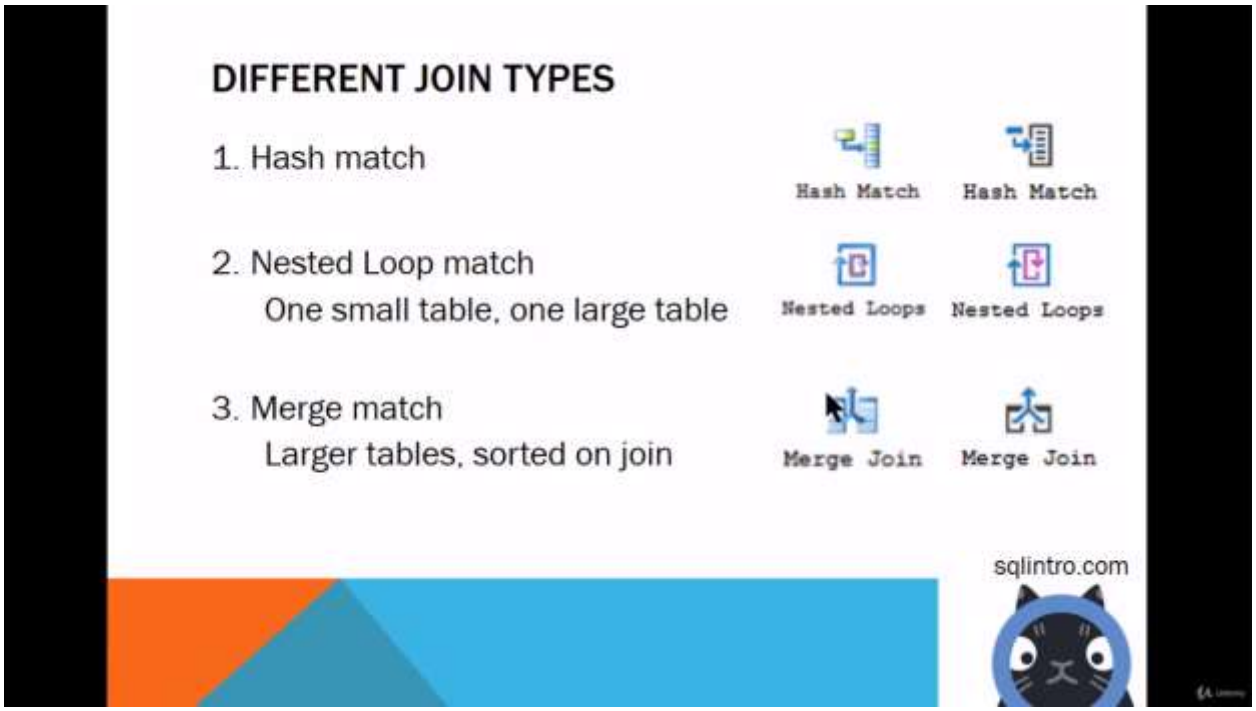

Select Name, DateOfBirth, CAST(DateOfBirth as Date) as [DatePart]
From Employees
Where DAY(DateOfBirth) = 9 AND MONTH(DateOfBirth) = 10
Name DateOfBirth DatePart
1 David 2017-10-09 08:26:14.800 2017-10-09
2 Elsa 2017-10-09 09:40:18.900 2017-10-09

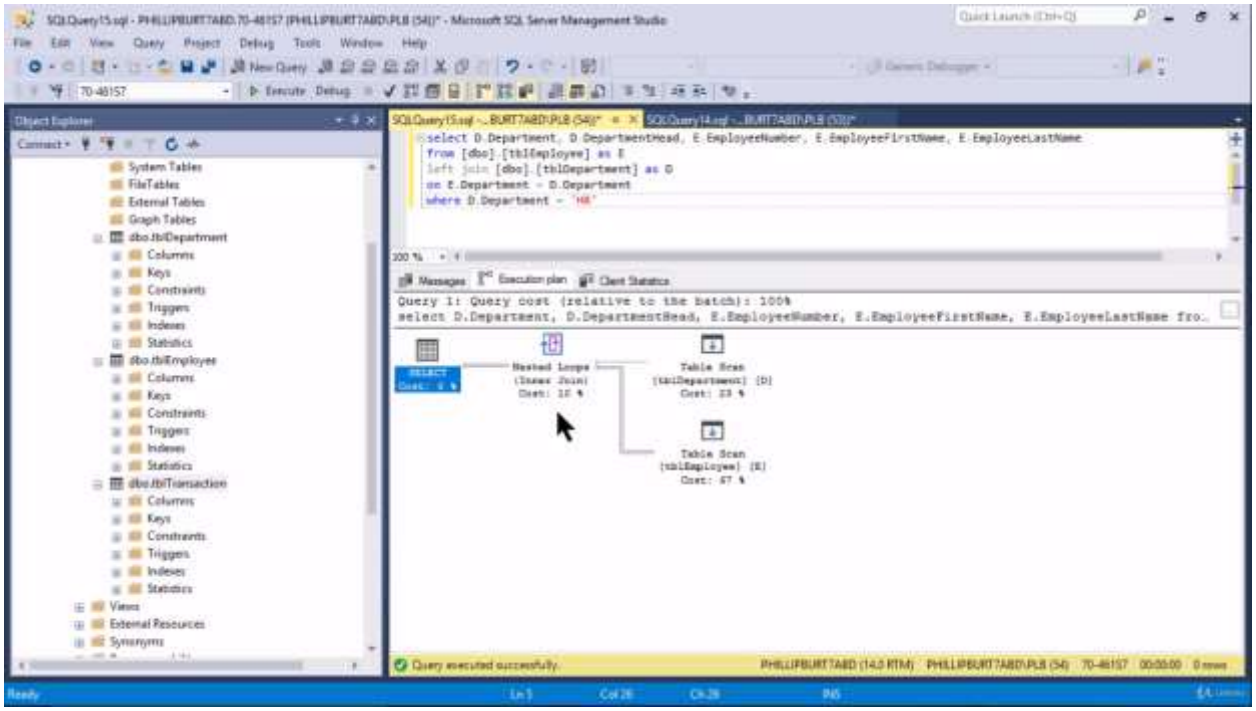

SQL Date Interview Questions
ID Name DateOfBirth
1 Tom 2018-11-19 10:36:46.520
2 Sara 2018-11-18 11:36:26.400
3 Bob 2017-12-22 10:40:10.300
4 Alex 2017-12-30 09:30:20.100
5 Charlie 2017-11-25 07:25:14.700
6 David 2017-10-09 08:26:14.800
7 Elsa 2017-10-09 09:40:18.900
8 George 2018-11-15 10:35:17.600
9 Mike 2018-11-16 09:14:17.600
10 Nancy 2018-11-17 11:16:18.600
Get all people who are born in a given year
(Example, all people born in the year 2017)
Name DateOfBirth DatePart
Bob 2017-12-22 10:40:10.300 2017-12-22
Alex 2017-12-30 09:30:20.100 2017-12-30
Charlie 2017-11-25 07:25:14.700 2017-11-25
David 2017-10-09 08:26:14.800 2017-10-09
Elsa 2017-10-09 09:40:18.900 2017-10-09
7
PRAGIM Technologies | 9900113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com

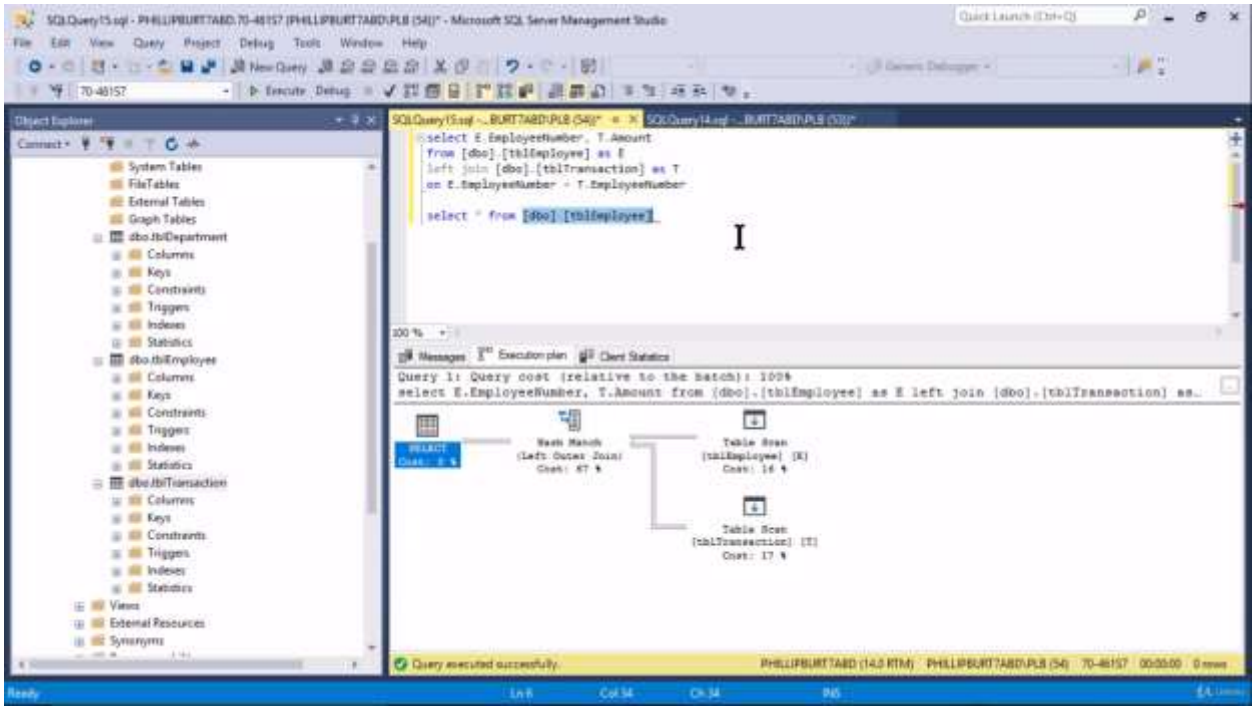

Select Name, DateOfBirth, CAST(DateOfBirth as Date) as [DatePart]
From Employees
Where CAST(DateOfBirth AS DATE) = DATEADD(DAY, -1, CAST(GETDATE() AS DATE))
Name DateOfBirth DatePart
1 Mike 2018-11-16 09:14:17.600 2018-11-16

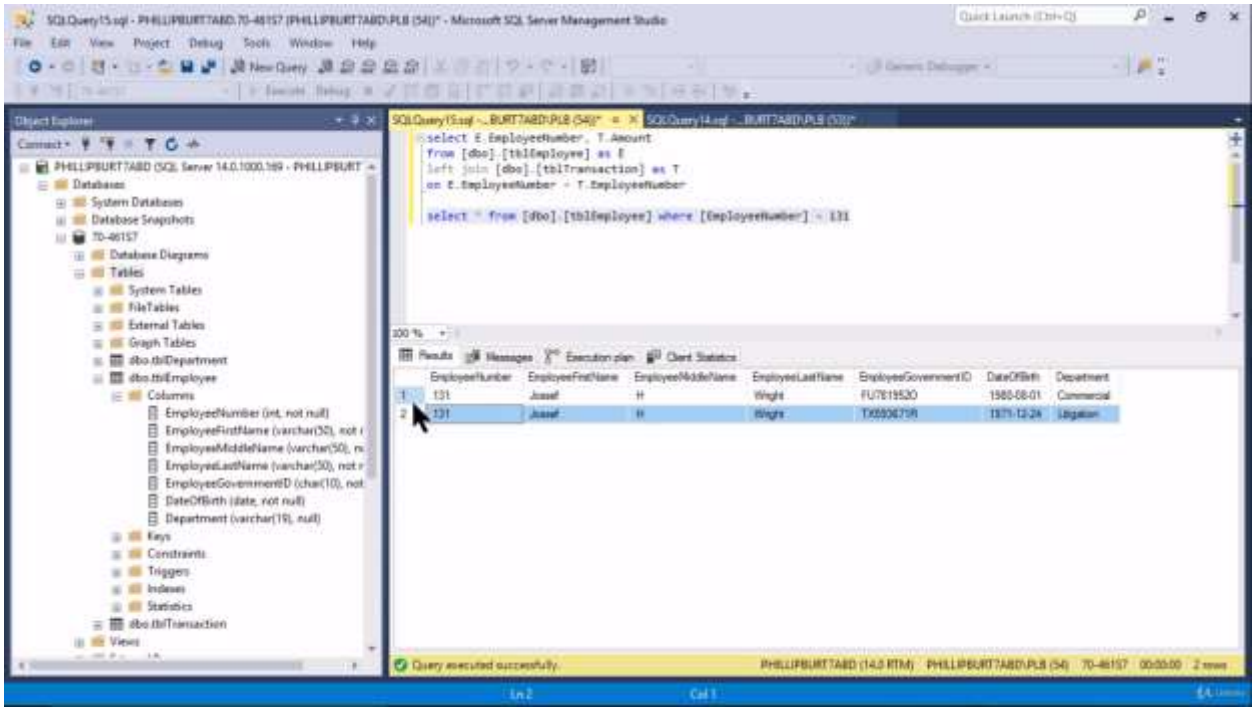

Select Name, DateOfBirth, CAST(DateOfBirth as Date) as [DatePart]
From Employees
Where CAST(DateOfBirth AS DATE)
BETWEEN DATEADD(DAY, -1, CAST(GETDATE() AS DATE))
AND CAST(GETDATE() AS DATE)
Name DateOfBirth DatePart
1 Mike 2018-11-16 09:14:17.600 2018-11-16
2 Nancy 2018-11-17 11:16:18.600 2018-11-17

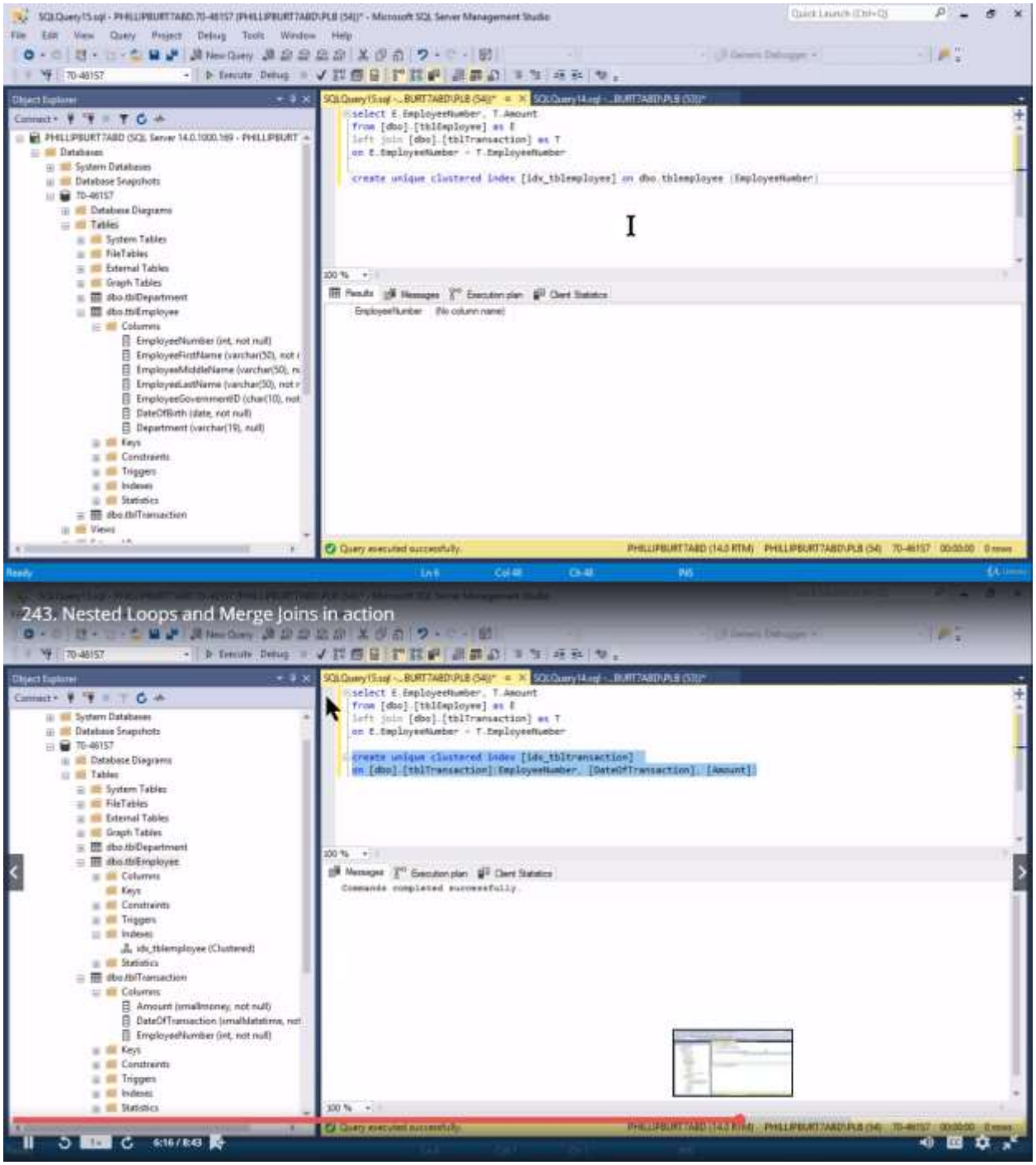

242. The different join types and seeing HASH join in action
select *
from [dbo].[tblEmployee] as E
left join [dbo].[tblDepartment] as D
on E.Department = D.Department
Query 1: Query cost (relative to the batch): 100%
select * from [dbo].[tblEmployee] as E left join [dbo].[tblDepartment] as D on E.Department = D.Department
SELECT
Cost: 0 %
Hash Match
(Right Outer Join)
Cost: 67 %
Table Scan
[tblDepartment] [D]
Cost: 9 %
Table Scan
[tblEmployee] [E]
Cost: 25 %

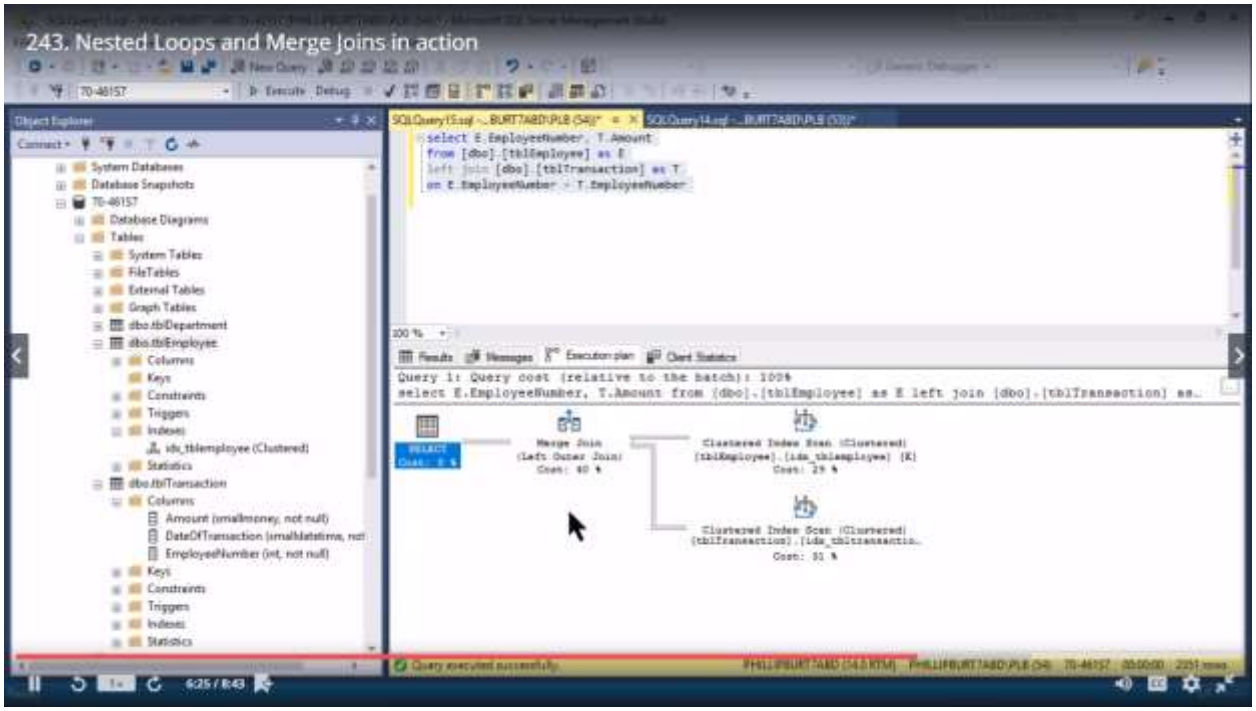

DIFFERENT JOIN TYPES
1. Hash match
Hash Match
Hash Match
2. Nested Loop match
One small table, one large table
Nested Loops
Nested Loops
3. Merge match
Larger tables, sorted on join
Merge Join
Merge Join
sqlintro.com

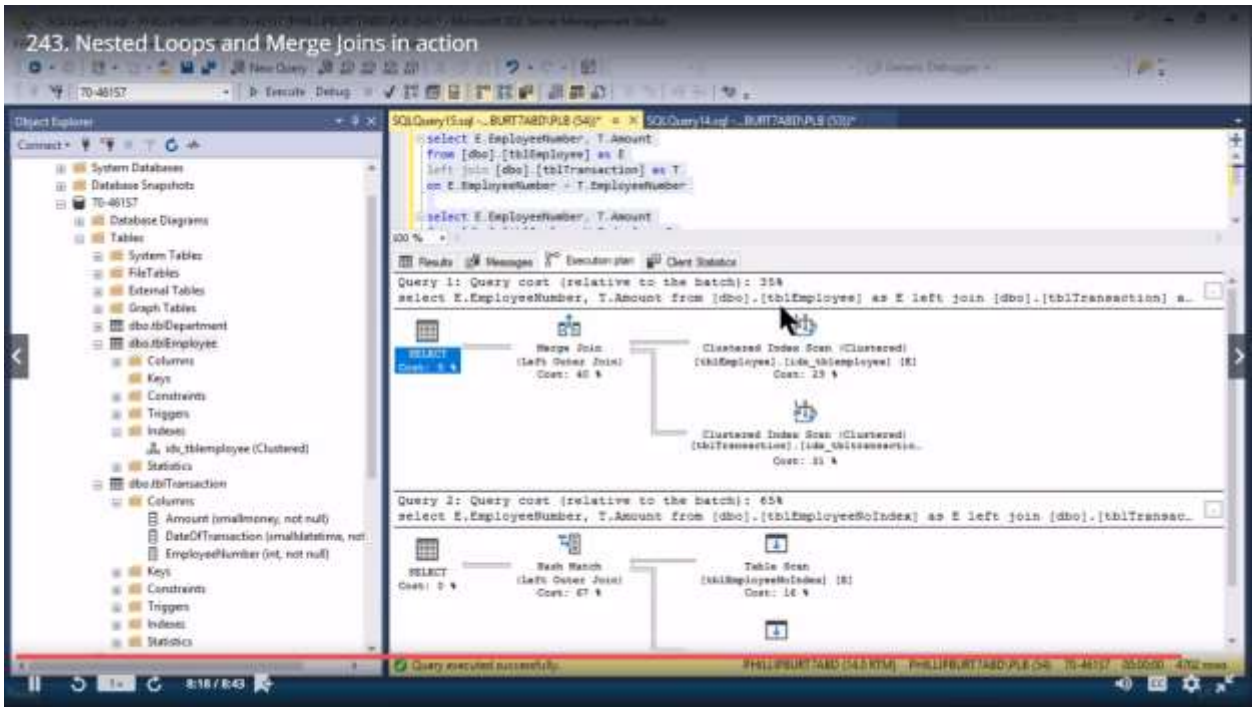

select D.Department, D.DepartmentHead, E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName
from [dbo].[tblEmployee] as E
left join [dbo].[tblDepartment] as D
on E.Department = D.Department
where D.Department = 'HR'
Query 1: Query cost (relative to the batch): 100%
select D.Department, D.DepartmentHead, E.EmployeeNumber, E.EmployeeFirstName, E.EmployeeLastName fro..
SELECT
Cost: 0 %
Nested Loops
(Inner Join)
Cost: 10 %
Table Scan
[tblDepartment] [D]
Cost: 23 %
Table Scan
[tblEmployee] [E]
Cost: 67 %
Query executed successfully.


select E.EmployeeNumber, T.Amount
from [dbo].[tblEmployee] as E
left join [dbo].[tblTransaction] as T
on E.EmployeeNumber = T.EmployeeNumber
select * from [dbo].[tblEmployee]
Query 1: Query cost (relative to the batch): 100%
select E.EmployeeNumber, T.Amount from [dbo].[tblEmployee] as E left join [dbo].[tblTransaction] as..
SELECT
Cost: 0 %
Hash Match
(Left Outer Join)
Cost: 67 %
Table Scan
[tblEmployee] [E]
Cost: 16 %
Table Scan
[tblTransaction] [T]
Cost: 17 %
Query executed successfully.


SQLQuery15.sql - PHILLIPBURT7ABD.70-46157 (PHILLIPBURT7ABD\PLB (54))* - Microsoft SQL Server Management Studio
File Edit View Project Debug Tools Window Help
New Query
Object Explorer
Connect
PHILLIPBURT7ABD (SQL Server 14.0.1000.169 - PHILLIPBURT
Databases
System Databases
Database Snapshots
70-46157
Database Diagrams
Tables
System Tables
FileTables
External Tables
Graph Tables
dbo.tblDepartment
dbo.tblEmployee
Columns
EmployeeNumber (int, not null)
EmployeeFirstName (varchar(30), not
EmployeeMiddleName (varchar(50), n
EmployeeLastName (varchar(50), not
EmployeeGovernmentID (char(10), not
DateOfBirth (date, not null)
Department (varchar(19), null)
Keys
Constraints
Triggers
Indexes
Statistics
dbo.tblTransaction
Views
SQLQuery15.sql -...BURT7ABD\PLB (54))*
SQLQuery14.sql -...BURT7ABD\PLB (53))*
select E.EmployeeNumber, T.Amount
from [dbo].[tblEmployee] as E
left join [dbo].[tblTransaction] as T
on E.EmployeeNumber = T.EmployeeNumber
select * from [dbo].[tblEmployee] where [EmployeeNumber] = 131
100 %
Results
Messages
Execution plan
Client Statistics
EmployeeNumber
EmployeeFirstName
EmployeeMiddleName
EmployeeLastName
EmployeeGovernmentID
DateOfBirth
Department
1 131 Josef H Wight FU7619520 1980-08-01 Commercial
2 131 Josef H Wight TX6936719R 1971-12-24 Litigation
Query executed successfully.
PHILLIPBURT7ABD (14.0 RTM)
PHILLIPBURT7ABD\PLB (54)
70-46157
00:00:00
2 rows
Ln 2
Col 1


243. Nested Loops and Merge Joins in action
select E.EmployeeNumber, T.Amount
from [dbo].[tblEmployee] as E
left join [dbo].[tblTransaction] as T
on E.EmployeeNumber = T.EmployeeNumber
create unique clustered index [idx_tblemployee] on dbo.tblemployee (EmployeeNumber)
create unique clustered index [idx_tbltransaction]
on [dbo].[tblTransaction] (EmployeeNumber, [DateOfTransaction], [Amount])
Commands completed successfully.
Query executed successfully.
5:16 / 8:43


243. Nested Loops and Merge Joins in action
select E.EmployeeNumber, T.Amount
from [dbo].[tblEmployee] as E
left join [dbo].[tblTransaction] as T
on E.EmployeeNumber = T.EmployeeNumber
Query 1: Query cost (relative to the batch): 100%
select E.EmployeeNumber, T.Amount from [dbo].[tblEmployee] as E left join [dbo].[tblTransaction] as...
SELECT
Cost: 0 %
Merge Join
(Left Outer Join)
Cost: 40 %
Clustered Index Scan (Clustered)
[tblEmployee].[idx_tblemployee] [E]
Cost: 29 %
Clustered Index Scan (Clustered)
[tblTransaction].[idx_tbltransactio...
Cost: 31 %
6:25 / 8:43


243. Nested Loops and Merge Joins in action
select E.EmployeeNumber, T.Amount
from [dbo].[tblEmployee] as E
left join [dbo].[tblTransaction] as T
on E.EmployeeNumber = T.EmployeeNumber
select E.EmployeeNumber, T.Amount
Query 1: Query cost (relative to the batch): 35%
select E.EmployeeNumber, T.Amount from [dbo].[tblEmployee] as E left join [dbo].[tblTransaction] as...
SELECT
Cost: 0 %
Merge Join
(Left Outer Join)
Cost: 40 %
Clustered Index Scan (Clustered)
[tblEmployee].[idx_tblemployee] [E]
Cost: 29 %
Clustered Index Scan (Clustered)
[tblTransaction].[idx_tbltransactio...
Cost: 31 %
Query 2: Query cost (relative to the batch): 65%
select E.EmployeeNumber, T.Amount from [dbo].[tblEmployeeNoIndex] as E left join [dbo].[tblTransac...
SELECT
Cost: 0 %
Hash Match
(Left Outer Join)
Cost: 67 %
Table Scan
[tblEmployeeNoIndex] [E]
Cost: 16 %
8:18 / 8:43

SARG – SEARCH ARGUMENTS

SARG can use Indexes.

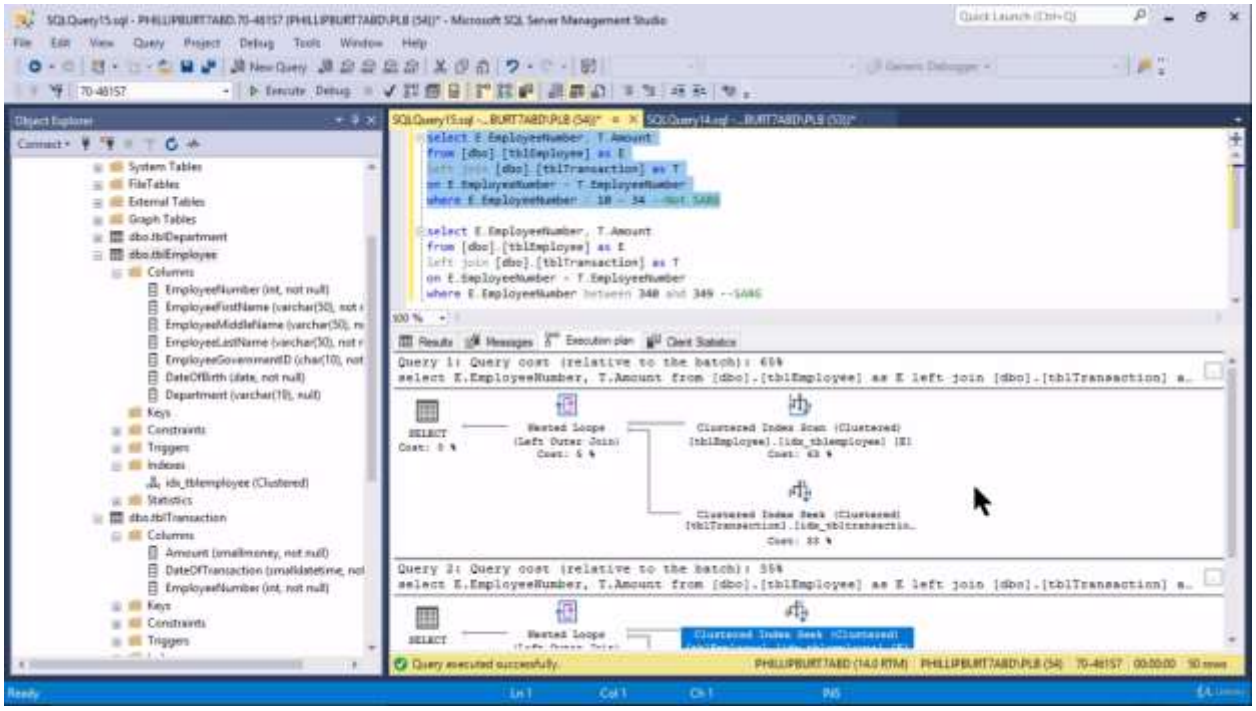
WHERE YEAR(DateOriginal) = 2020

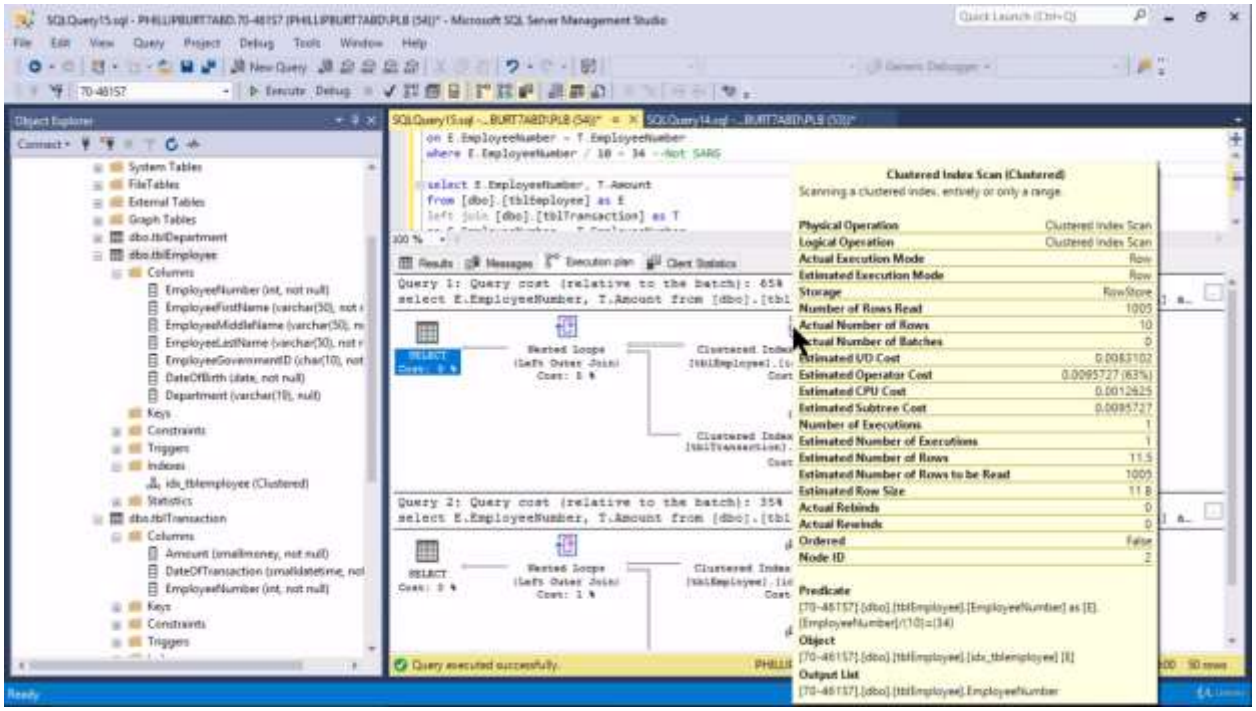
WHERE DateOriginal >= '2020-01-01' AND DateOriginal < '2021-01-01'

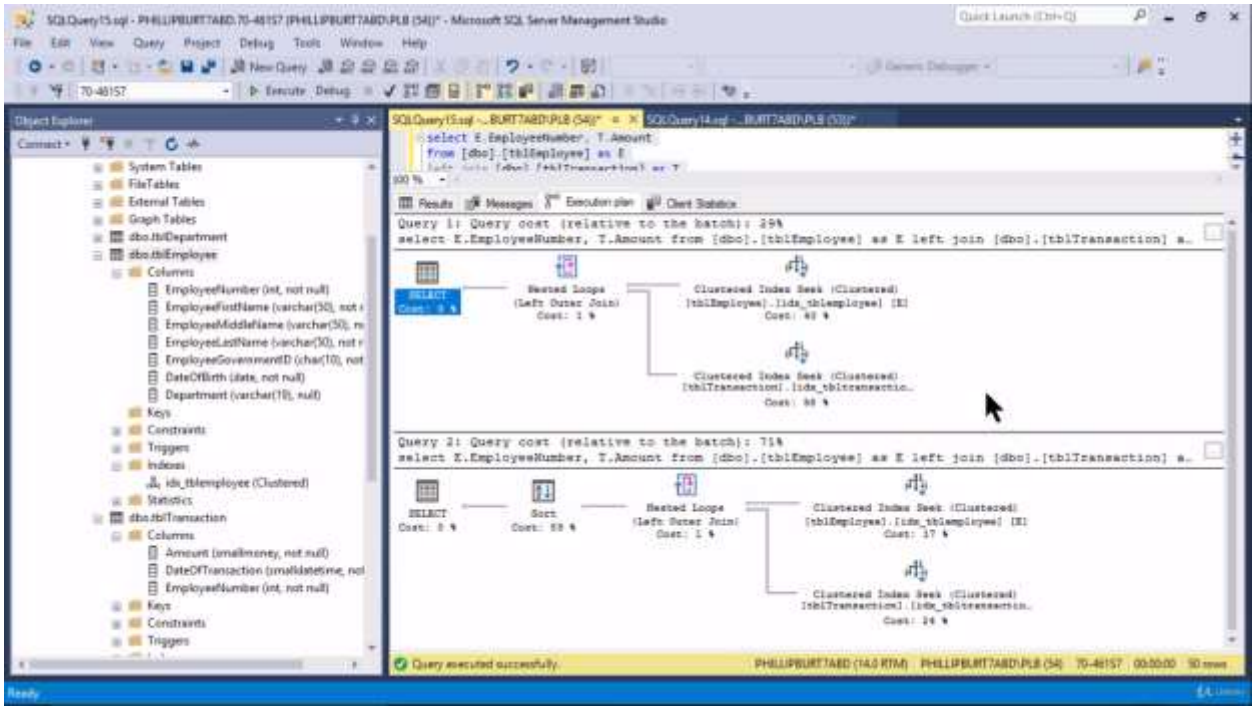
WHERE SUBSTRING(PersonName,1,4) = 'John'

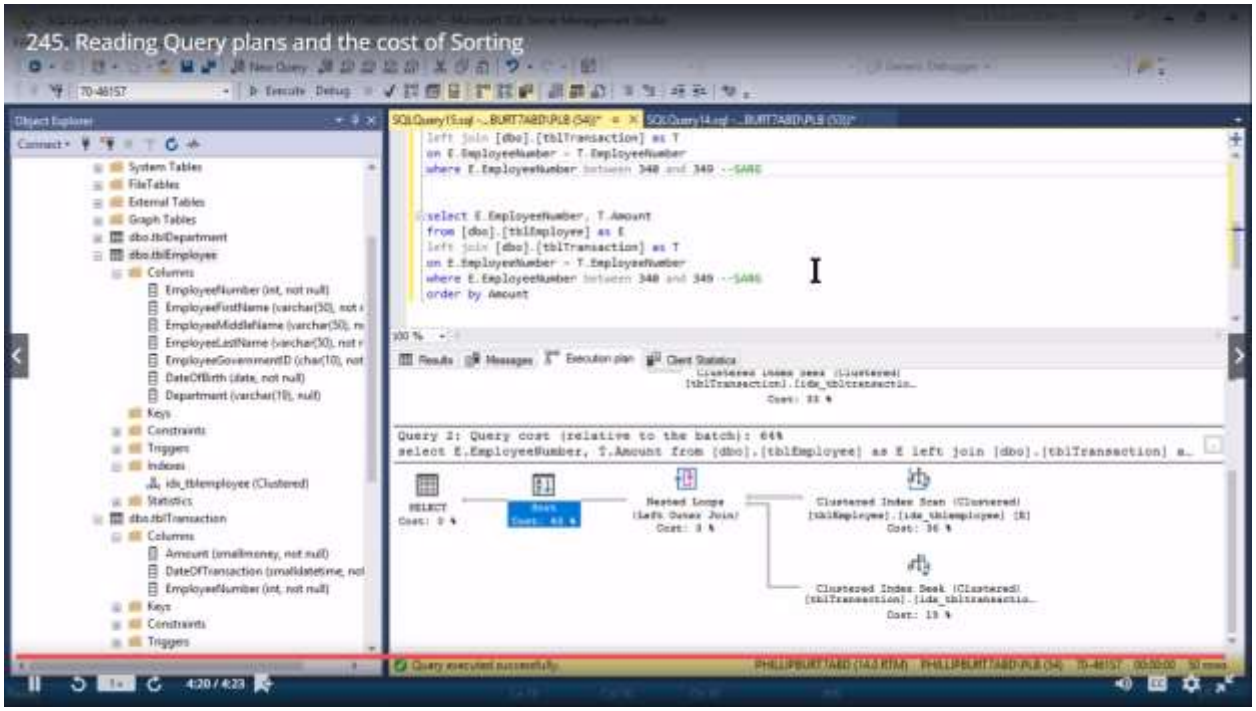
WHERE PersonName Like 'John%'

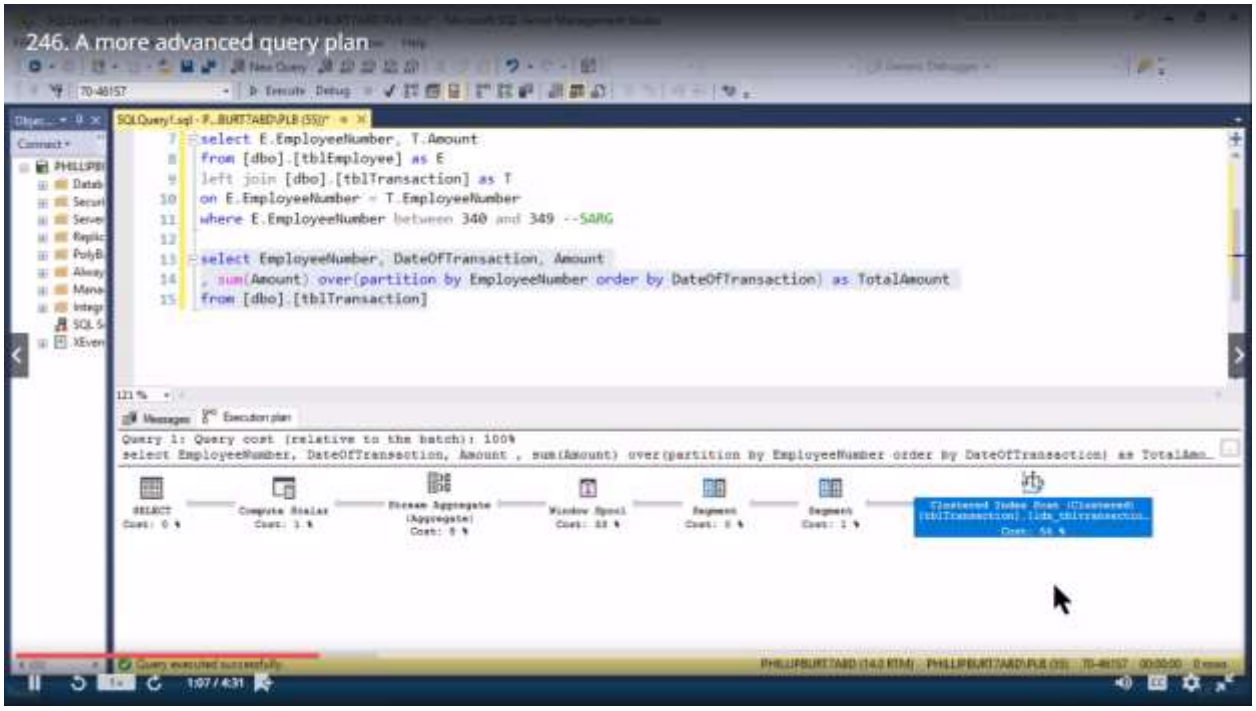
sqlintro.com




Clustered Index Scan (Clustered)
Scanning a clustered index, entirely or only a range.
Physical Operation
Clustered Index Scan
Logical Operation
Clustered Index Scan
Actual Execution Mode
Row
Estimated Execution Mode
Row
Storage
RowStore
Number of Rows Read
1005
Actual Number of Rows
10
Actual Number of Batches
0
Estimated I/O Cost
0.0063102
Estimated Operator Cost
0.0095727 (63%)
Estimated CPU Cost
0.0012625
Estimated Subtree Cost
0.0095727
Number of Executions
1
Estimated Number of Executions
1
Estimated Number of Rows
11.5
Estimated Number of Rows to be Read
1005
Estimated Row Size
11 B
Actual Rebinds
0
Actual Rewinds
0
Ordered
False
Node ID
2
Predicate
[70-46157].[dbo].[tblEmployee].[EmployeeNumber] as [E].[EmployeeNumber]/(10)<(34)
Object
[70-46157].[dbo].[tblEmployee].[idx_tblemployee] [E]
Output List
[70-46157].[dbo].[tblEmployee].EmployeeNumber




245. Reading Query plans and the cost of Sorting
select E.EmployeeNumber, T.Amount
from [dbo].[tblEmployee] as E
left join [dbo].[tblTransaction] as T
on E.EmployeeNumber = T.EmployeeNumber
where E.EmployeeNumber between 340 and 349 --SARG
order by Amount
Query 2: Query cost (relative to the batch): 64%
SELECT
Nested Loops
(Left Outer Join)
Clustered Index Scan (Clustered)
Clustered Index Seek (Clustered)
4:20 / 4:23

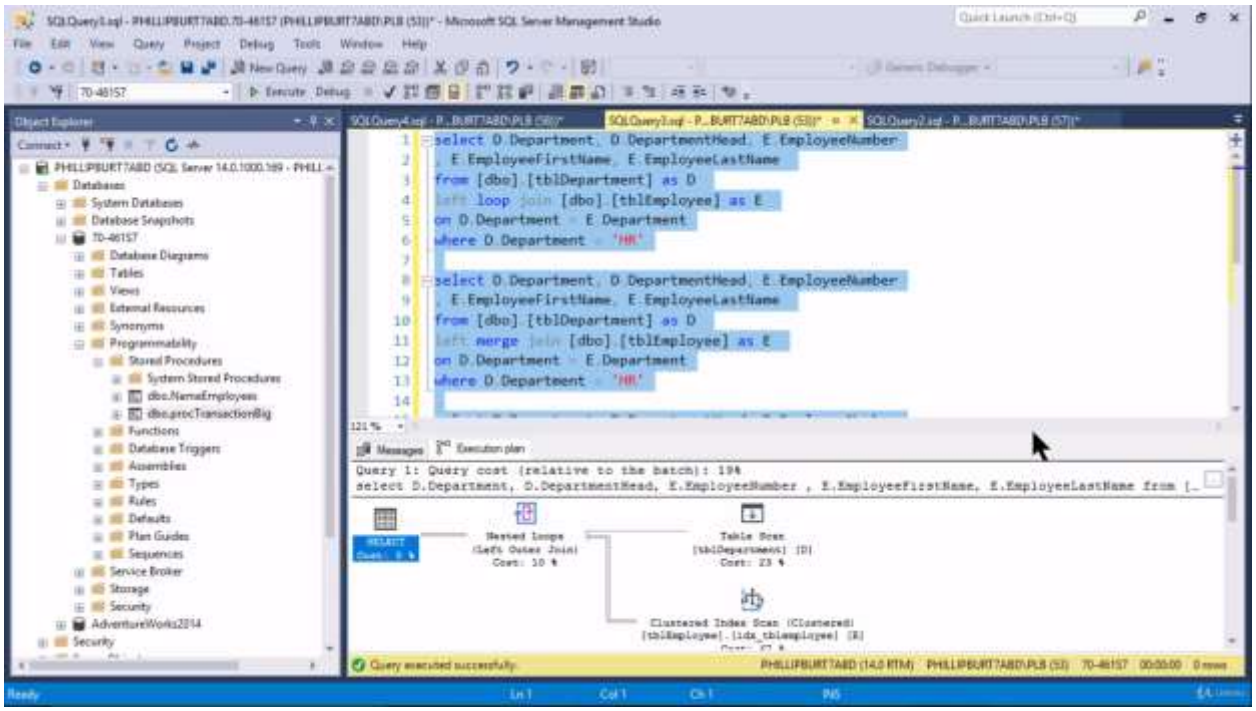

246. A more advanced query plan
select E.EmployeeNumber, T.Amount
from [dbo].[tblEmployee] as E
left join [dbo].[tblTransaction] as T
on E.EmployeeNumber = T.EmployeeNumber
where E.EmployeeNumber between 340 and 349 --SARG
select EmployeeNumber, DateOfTransaction, Amount
, sum(Amount) over(partition by EmployeeNumber order by DateOfTransaction) as TotalAmount
from [dbo].[tblTransaction]
Query 1: Query cost (relative to the batch): 100%
SELECT
Compute Scalar
Stream Aggregate
(Aggregate)
Window Spool
Segment
Segment
1:07 / 4:31

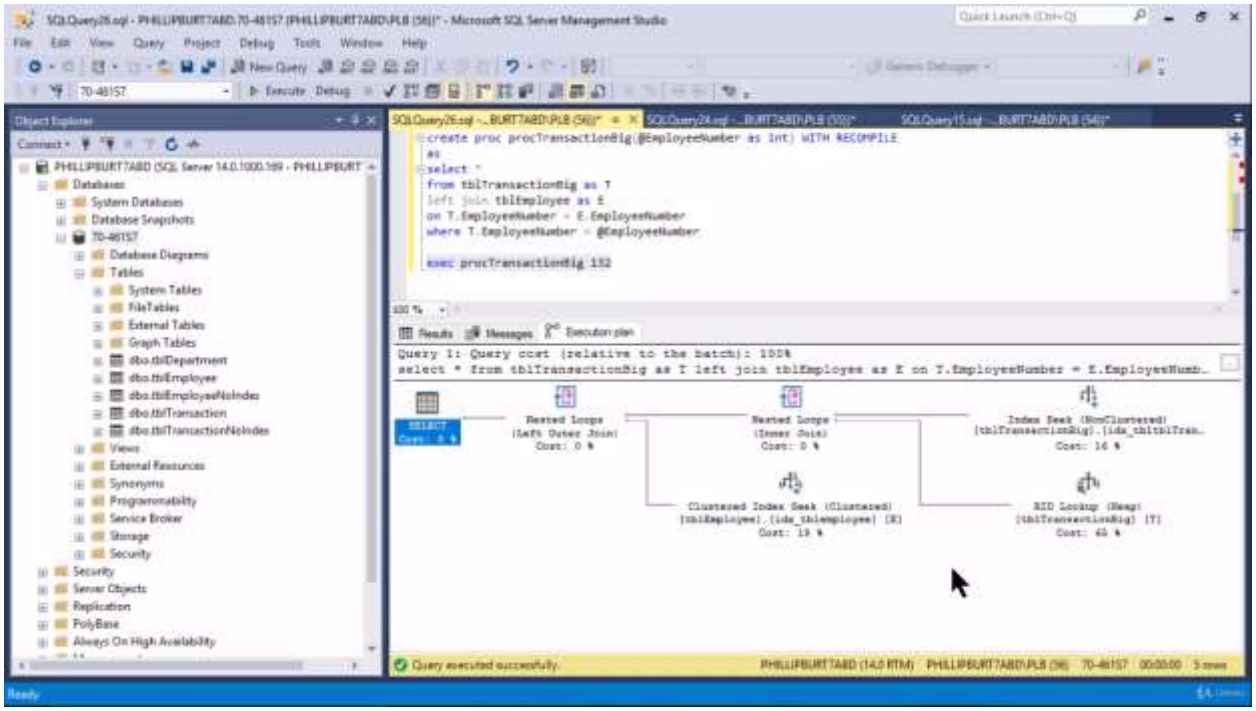
HINTS

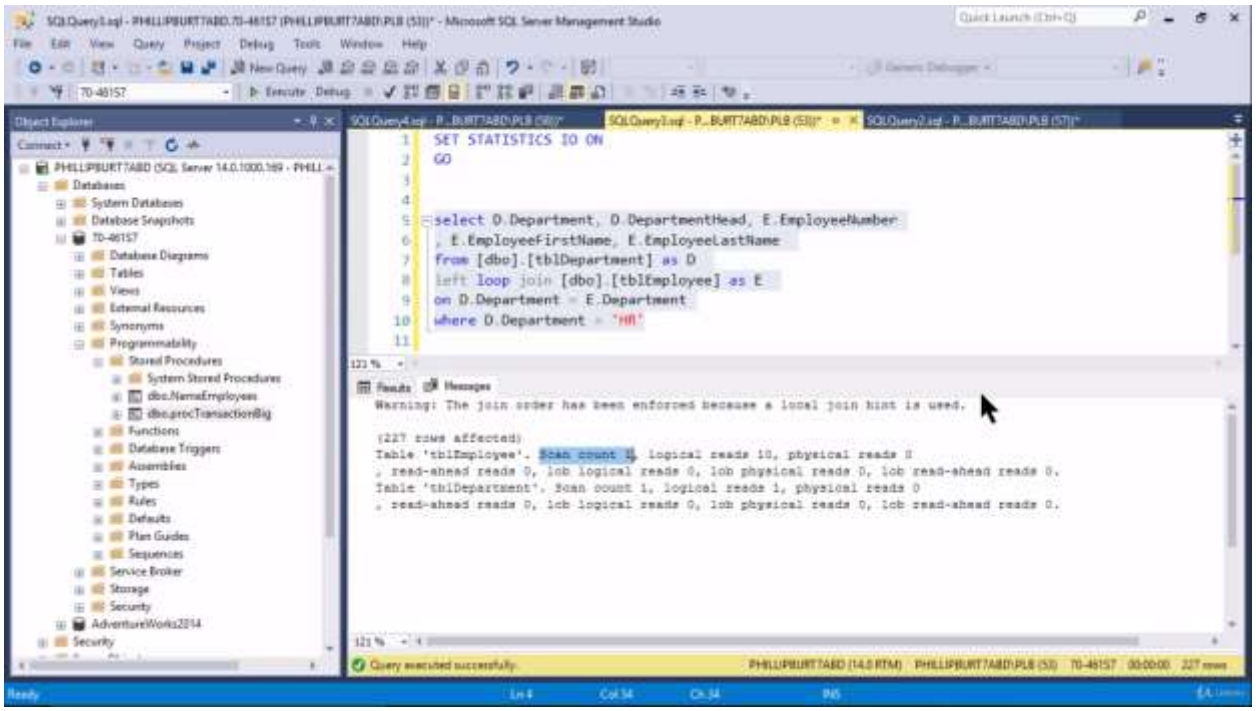
WITH

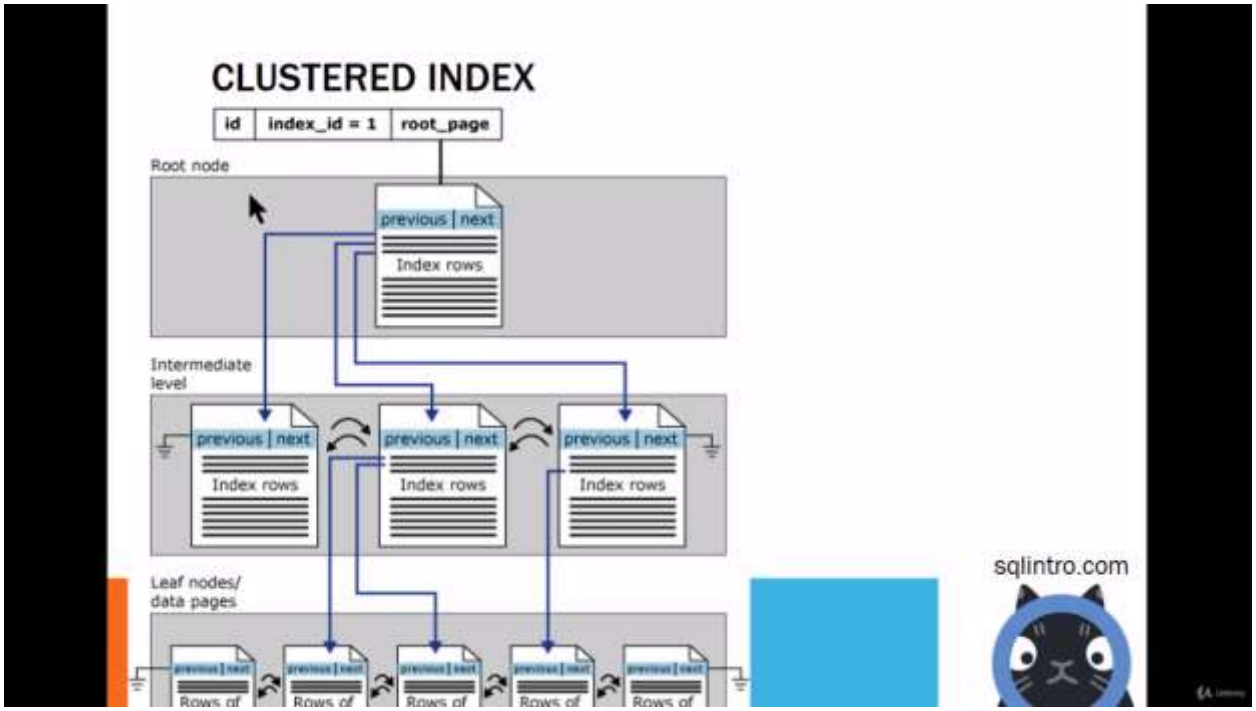
- NOLOCK
- READUNCOMMITTED
- UPDLOCK
- REPEATABLEREAD
- SERIALIZABLE
- READCOMMITTED
- TABLOCK
- TABLOCKX
- PAGLOCK
- ROWLOCK
- NOWAIT
- READPAST
- XLOCK
- SNAPSHOT
- NOEXPAND
- FORCESEEK
- FORCESCAN

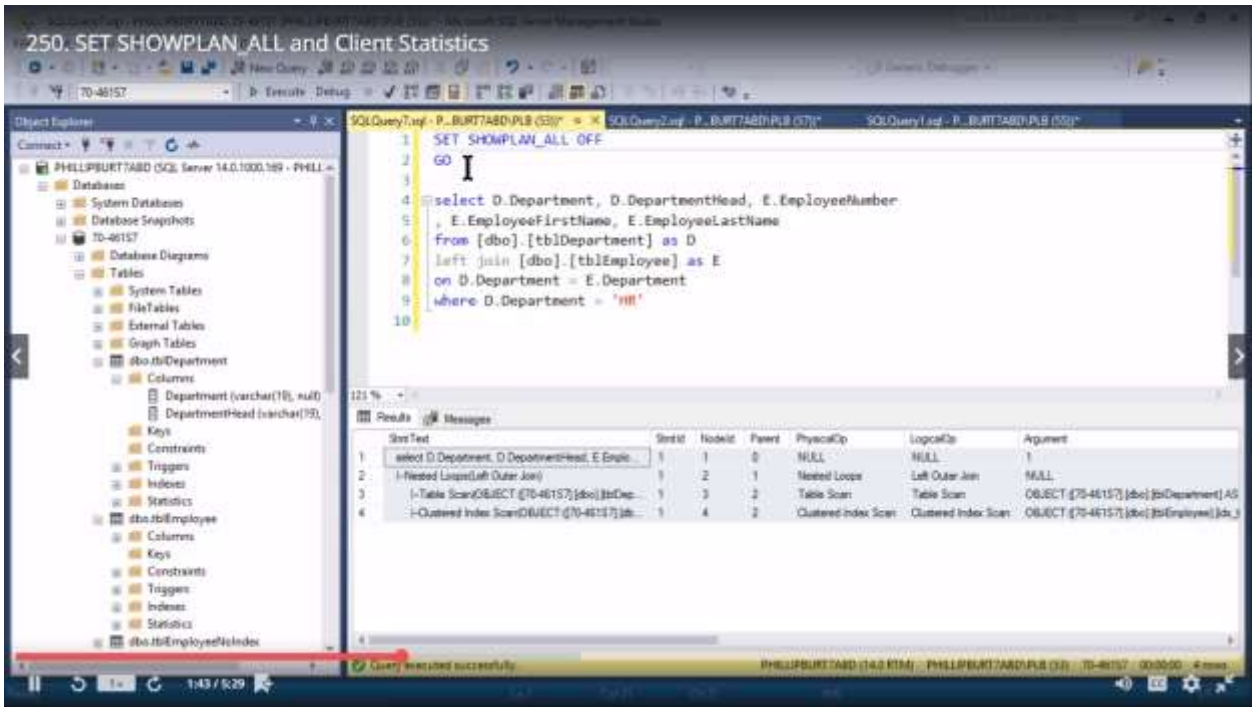
sqlintro.com

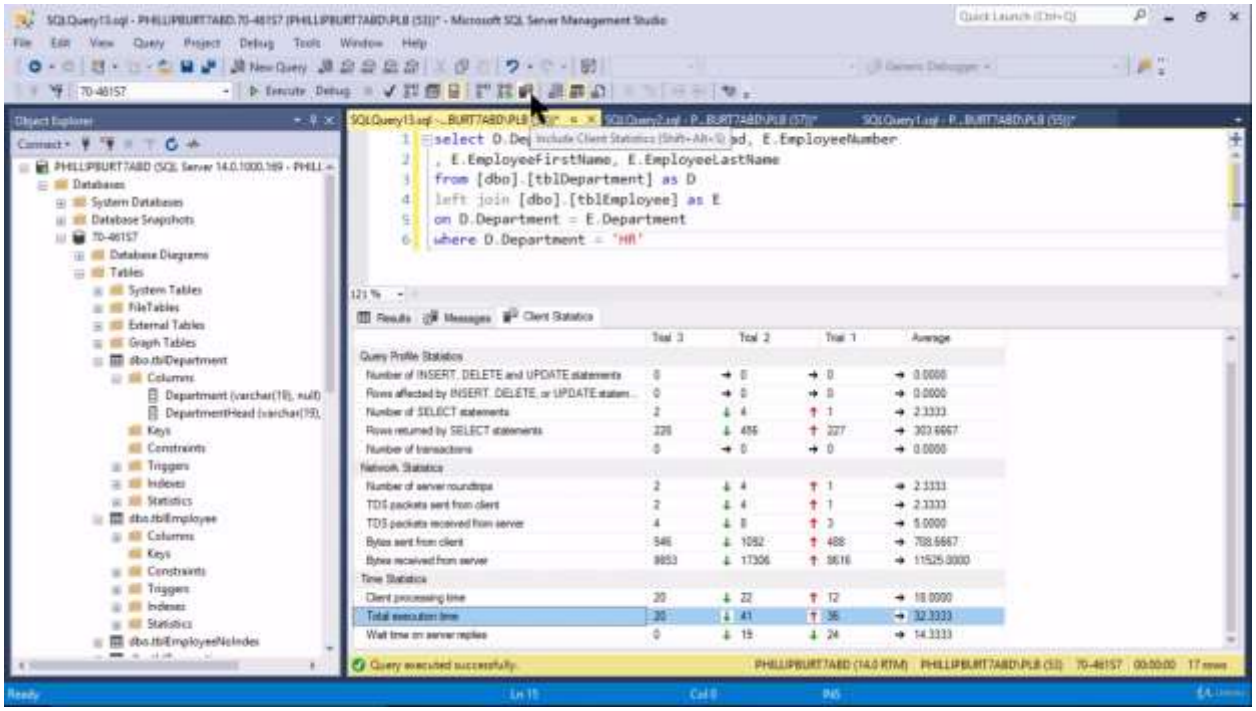


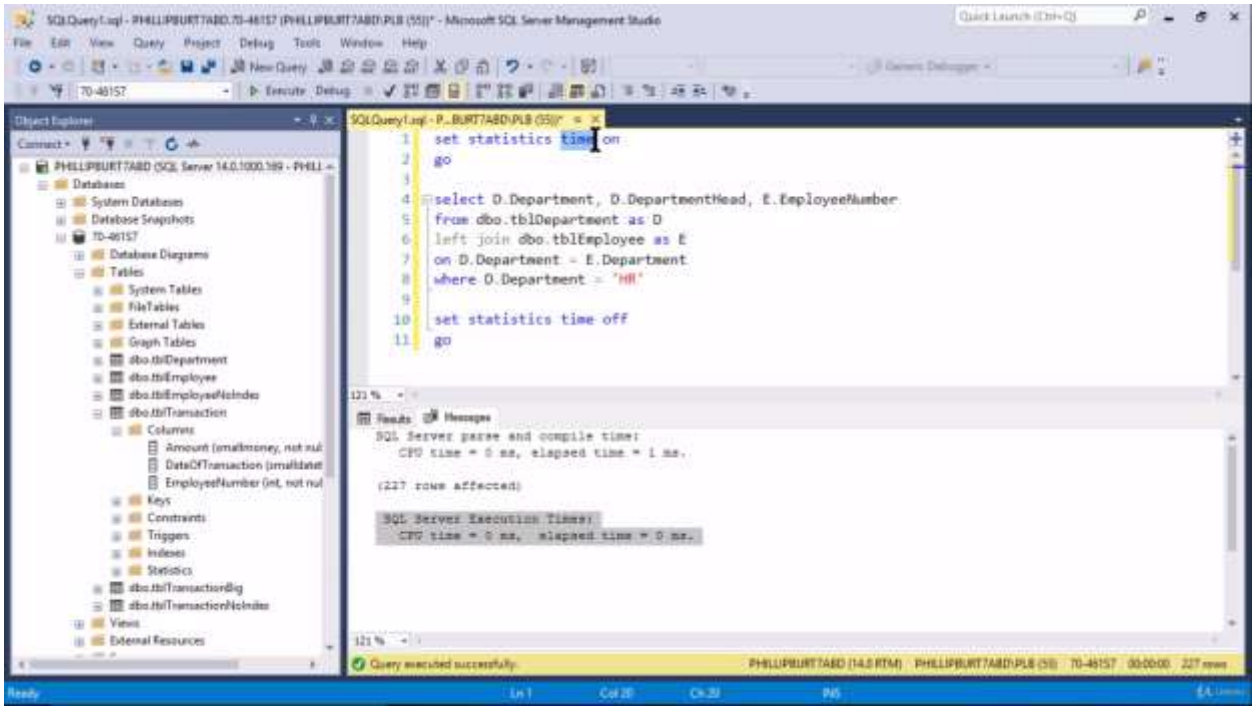

create proc procTransactionBig(@EmployeeNumber as int) WITH RECOMPILE
as
select *
from tblTransactionBig as T
left join tblEmployee as E
on T.EmployeeNumber = E.EmployeeNumber
where T.EmployeeNumber = @EmployeeNumber
exec procTransactionBig 152
Query executed successfully.

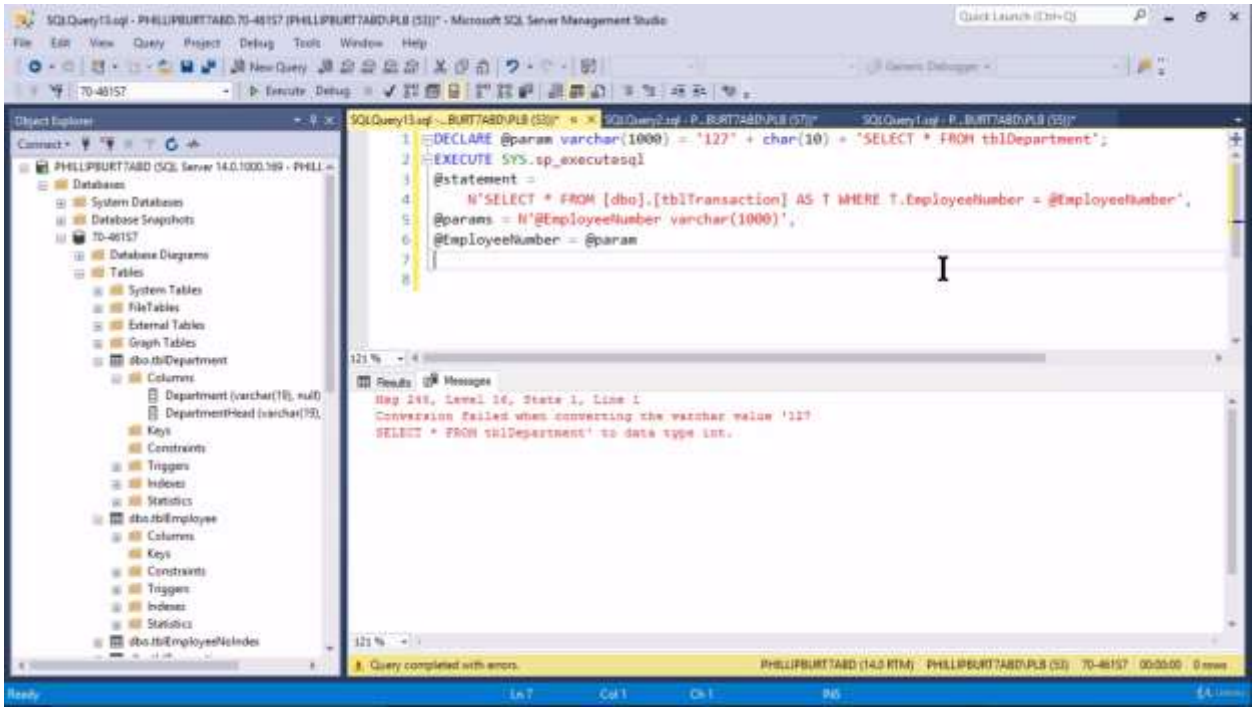

SET STATISTICS IO ON
GO
select D.Department, D.DepartmentHead, E.EmployeeNumber
, E.EmployeeFirstName, E.EmployeeLastName
from [dbo].[tblDepartment] as D
left loop join [dbo].[tblEmployee] as E
on D.Department = E.Department
where D.Department = 'HR'
Warning: The join order has been enforced because a local join hint is used.
(227 rows affected)
Table 'tblEmployee'. Scan count 1, logical reads 10, physical reads 0
, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Table 'tblDepartment'. Scan count 1, logical reads 1, physical reads 0
, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Query executed successfully.


CLUSTERED INDEX
id
index_id = 1
root_page
Root node
previous | next
Index rows
Intermediate level
previous | next
Index rows
previous | next
Index rows
previous | next
Index rows
Leaf nodes/ data pages
Rows of
Rows of
Rows of
Rows of
Rows of
sqlintro.com


250. SET SHOWPLAN_ALL and Client Statistics
SET SHOWPLAN_ALL OFF
GO
select D.Department, D.DepartmentHead, E.EmployeeNumber
, E.EmployeeFirstName, E.EmployeeLastName
from [dbo].[tblDepartment] as D
left join [dbo].[tblEmployee] as E
on D.Department = E.Department
where D.Department = 'HR'
Results
Messages
1:43 / 5:29


select D.Department, D.DepartmentHead, E.EmployeeNumber
, E.EmployeeFirstName, E.EmployeeLastName
from [dbo].[tblDepartment] as D
left join [dbo].[tblEmployee] as E
on D.Department = E.Department
where D.Department = 'HR'
Results
Messages
Client Statistics
Trial 3
Trial 2
Trial 1
Average
Query Profile Statistics
Number of INSERT, DELETE and UPDATE statements
Rows affected by INSERT, DELETE, or UPDATE statem...
Number of SELECT statements
Rows returned by SELECT statements
Number of transactions
Network Statistics
Number of server roundtrips
TDS packets sent from client
TDS packets received from server
Bytes sent from client
Bytes received from server
Time Statistics
Client processing time
Total execution time
Wait time on server replies
Query executed successfully.

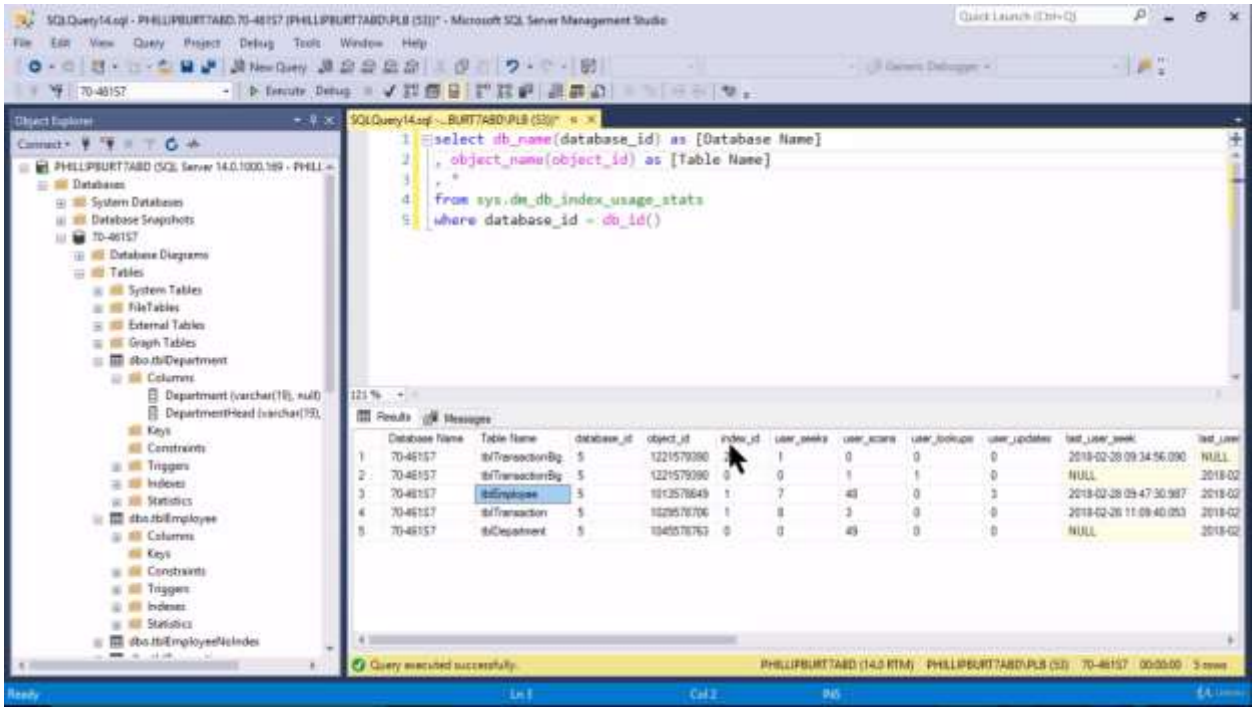

set statistics time on
go
select D.Department, D.DepartmentHead, E.EmployeeNumber
from dbo.tblDepartment as D
left join dbo.tblEmployee as E
on D.Department = E.Department
where D.Department = 'HR'
set statistics time off
go
Results
Messages
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 1 ms.
(227 rows affected)
SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 0 ms.
Query executed successfully.

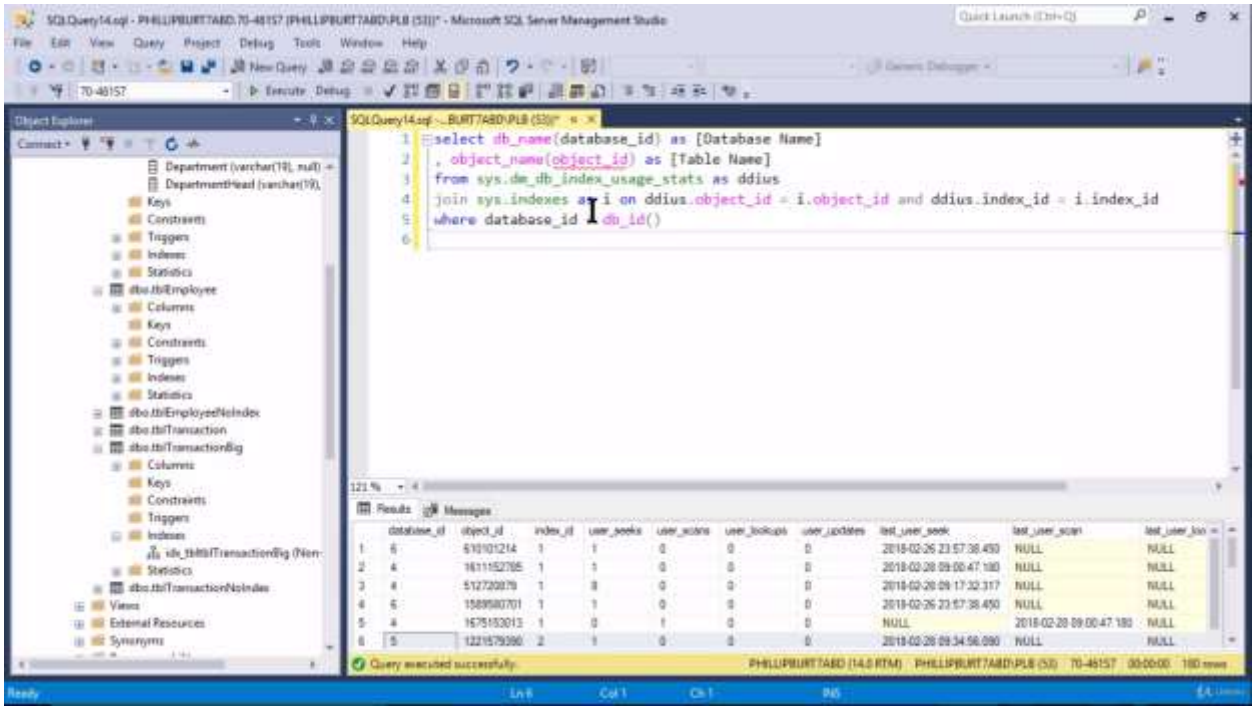


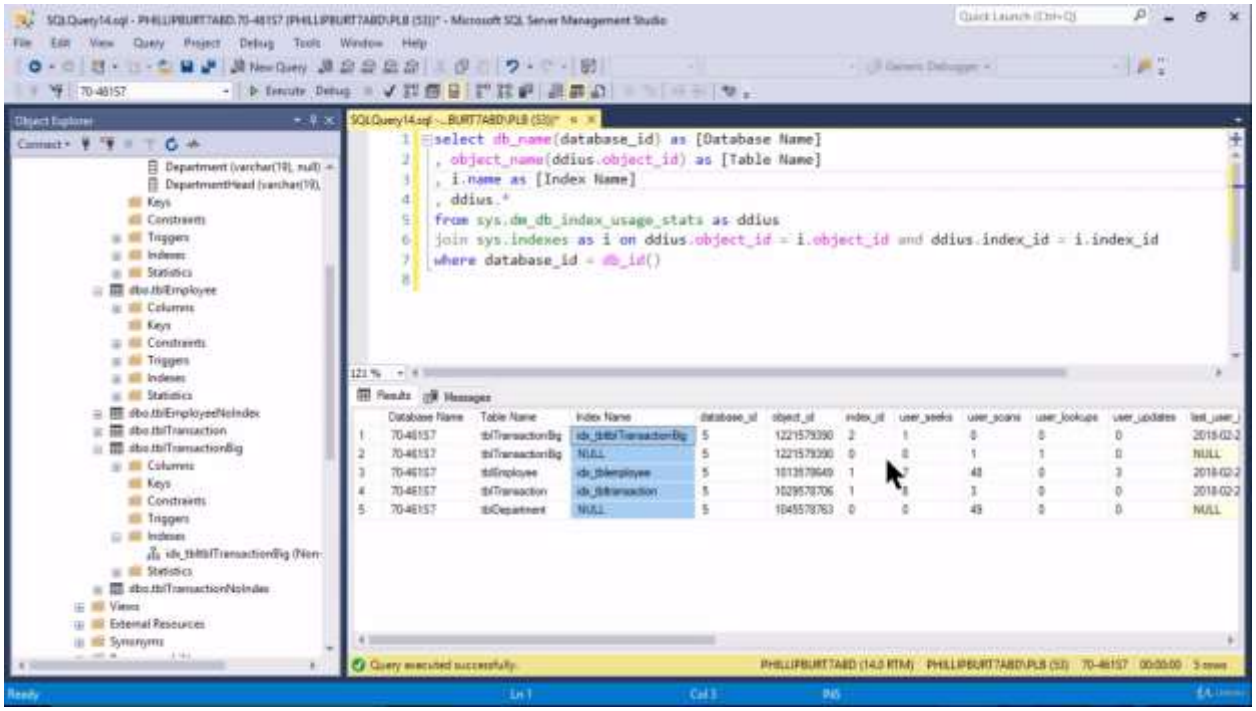
DMVS (INDEX RELATED DYNAMIC MANAGEMENT VIEWS AND FUNCTIONS)

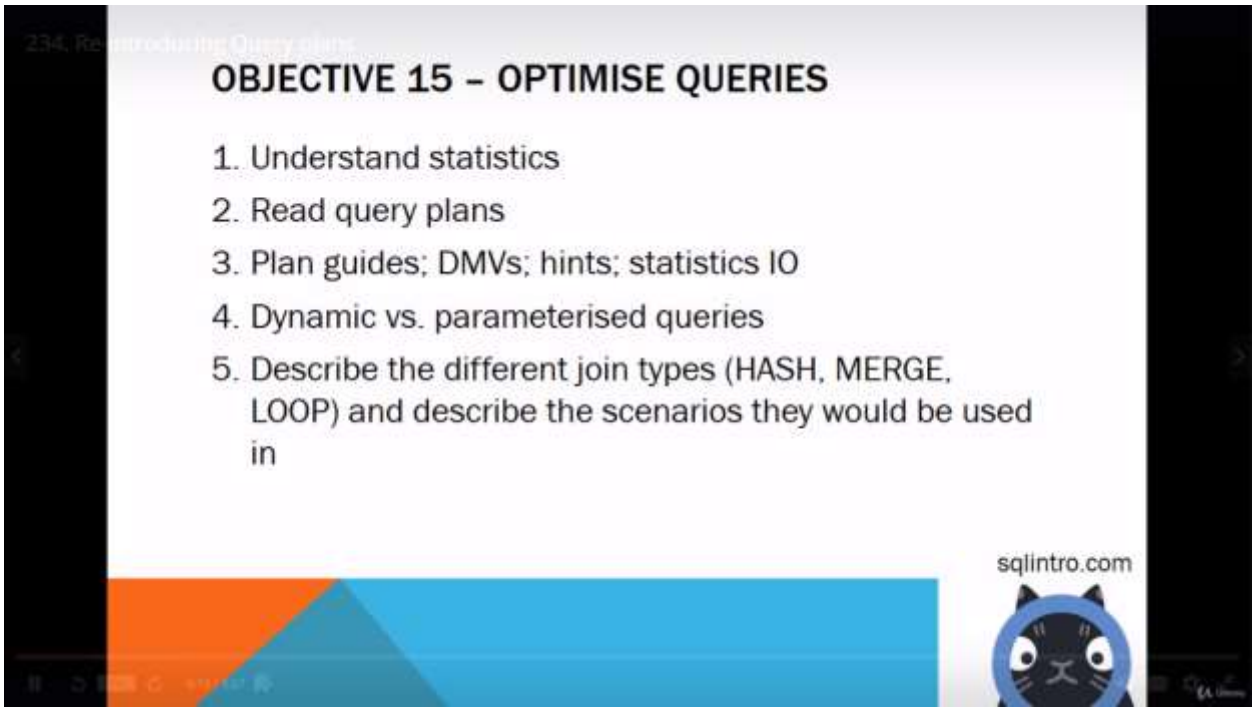
- sys.dm_db_index_usage_stats
- sys.dm_db_missing_index_details
  - sys.dm_db_missing_index_columns
  - sys.dm_db_missing_index_groups
- sys.dm_db_index_physical_stats

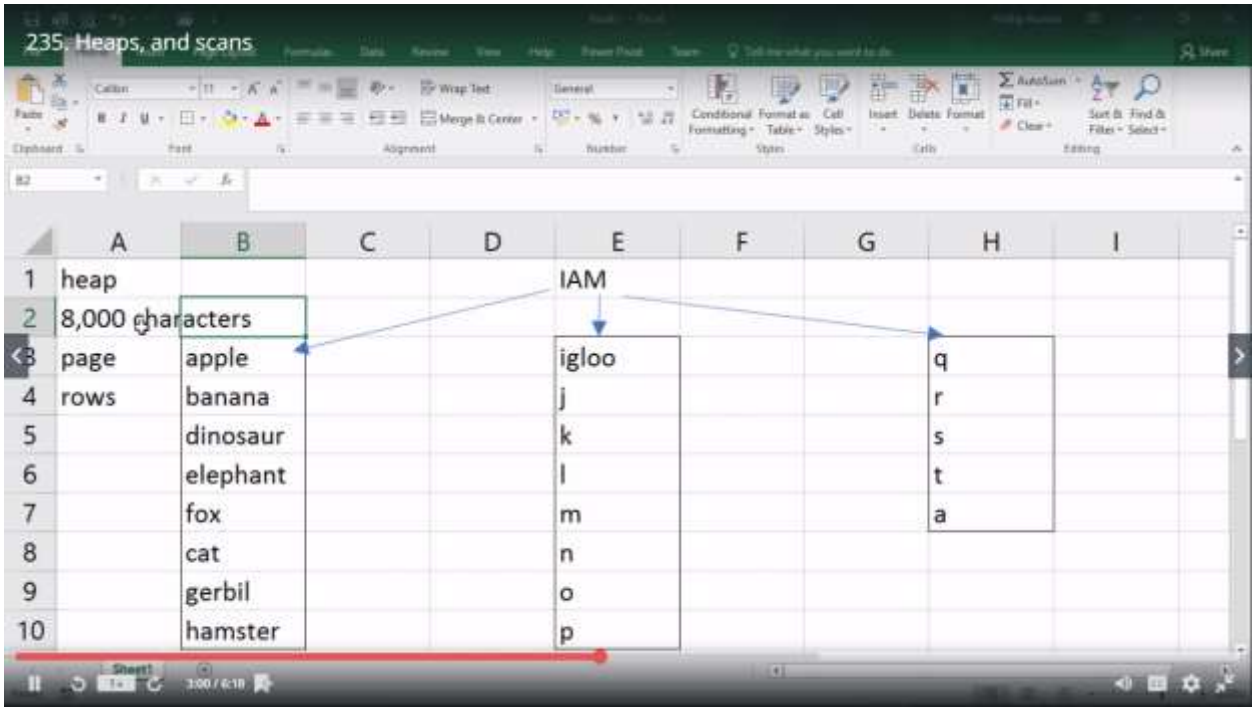
sqlintro.com

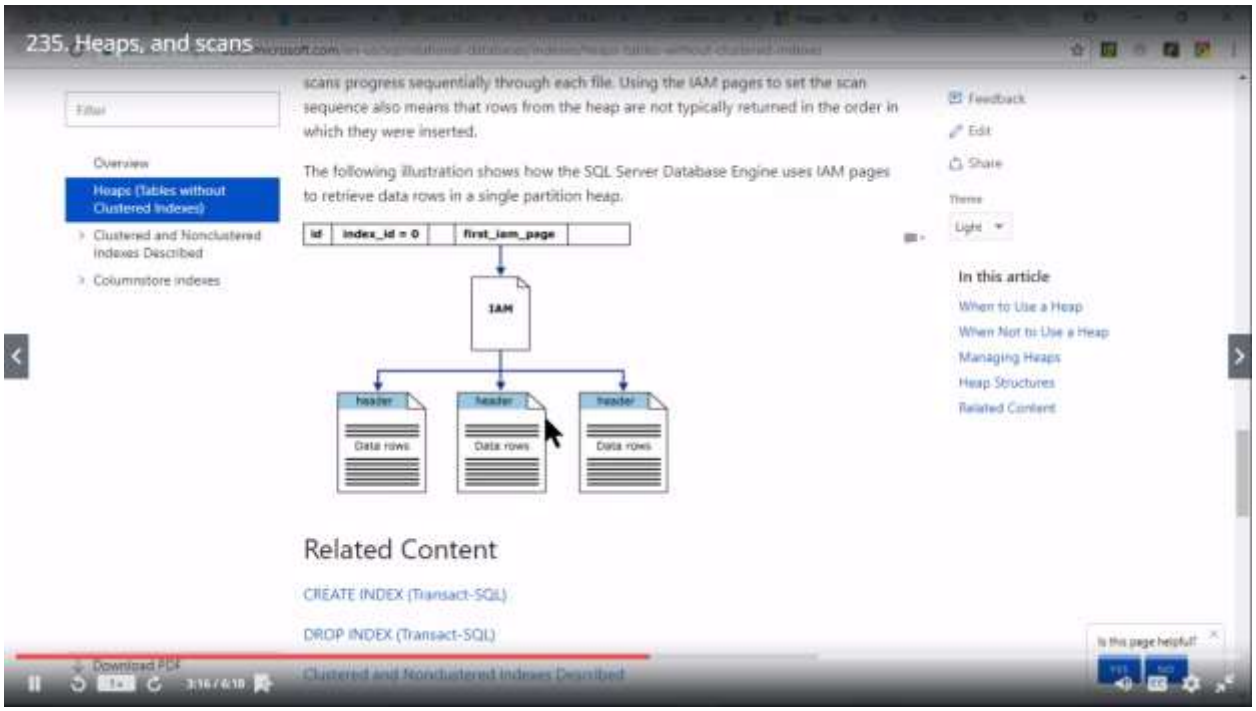

SQLQuery14.sql - PHILLIPBURT7ABD.70-461S7 (PHILLIPBURT7ABD\PLB (53))* - Microsoft SQL Server Management Studio
select db_name(database_id) as [Database Name]
, object_name(object_id) as [Table Name]
, *
from sys.dm_db_index_usage_stats
where database_id = db_id()
Database Name
Table Name
database_id
object_id
index_id
user_seeks
user_scans
user_lookups
user_updates
last_user_seek
70-461S7
tblTransactionBig
tblEmployee
tblTransaction
tblDepartment
Query executed successfully.


SQLQuery14.sql - PHILLIPBURT7ABD.70-461S7 (PHILLIPBURT7ABD\PLB (53))* - Microsoft SQL Server Management Studio
select db_name(database_id) as [Database Name]
, object_name(object_id) as [Table Name]
from sys.dm_db_index_usage_stats as ddius
join sys.indexes as i on ddius.object_id = i.object_id and ddius.index_id = i.index_id
where database_id = db_id()
database_id
object_id
index_id
user_seeks
user_scans
user_lookups
user_updates
last_user_seek
last_user_scan
Query executed successfully.


SQLQuery14.sql - PHILLIPBURT7ABD.70-46157 (PHILLIPBURT7ABD\PLB (53))* - Microsoft SQL Server Management Studio
Object Explorer
SQLQuery14.sql - ...BURT7ABD\PLB (53))*
select db_name(database_id) as [Database Name]
, object_name(ddius.object_id) as [Table Name]
, i.name as [Index Name]
, ddius.*
from sys.dm_db_index_usage_stats as ddius
join sys.indexes as i on ddius.object_id = i.object_id and ddius.index_id = i.index_id
where database_id = db_id()
Results
Messages
Database Name
Table Name
Index Name
Query executed successfully.


OBJECTIVE 15 – OPTIMISE QUERIES
1. Understand statistics
2. Read query plans
3. Plan guides; DMVs; hints; statistics IO
4. Dynamic vs. parameterised queries
5. Describe the different join types (HASH, MERGE, LOOP) and describe the scenarios they would be used in
sqlintro.com


235. Heaps, and scans
heap
8,000 characters
page
rows
apple
banana
dinosaur
elephant
fox
cat
gerbil
hamster
IAM
igloo
j
k
l
m
n
o
p
q
r
s
t
a


235. Heaps, and scans
scans progress sequentially through each file. Using the IAM pages to set the scan sequence also means that rows from the heap are not typically returned in the order in which they were inserted.
The following illustration shows how the SQL Server Database Engine uses IAM pages to retrieve data rows in a single partition heap.
id
index_id = 0
first_iam_page
IAM
header
Data rows
Overview
Heaps (Tables without Clustered Indexes)
Clustered and Nonclustered Indexes Described
Columnstore indexes
In this article
When to Use a Heap
When Not to Use a Heap
Managing Heaps
Heap Structures
Related Content
Related Content
CREATE INDEX (Transact-SQL)
DROP INDEX (Transact-SQL)
Clustered and Nonclustered Indexes Described

Searching is similar to searching a binary search tree. Starting at the root, the tree is recursively traversed from top to bottom. At each level, the search reduces its field of view to the child pointer (subtree) whose range includes the search value. A subtree's range is defined by the values, or keys, contained in its parent node. These limiting values are also known as separation values.

Binary search is typically (but not necessarily) used within nodes to find the separation values and child tree of interest.

Insertion [edit]

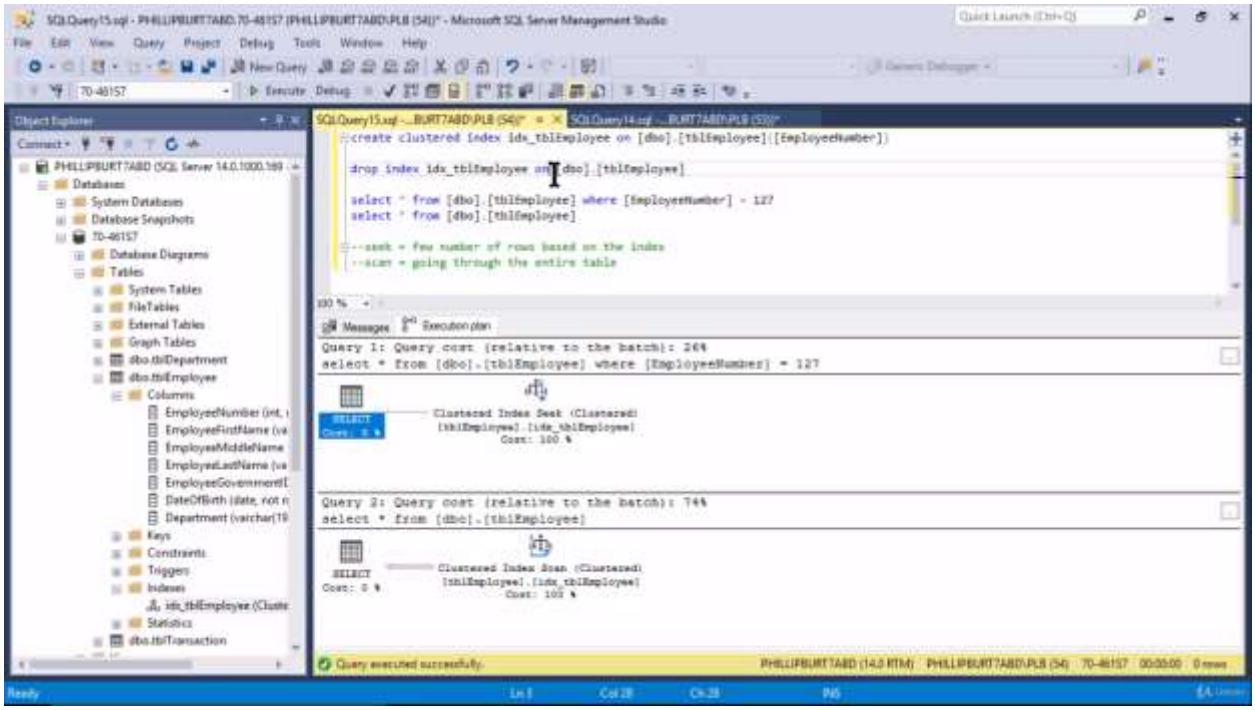
All insertions start at a leaf node. To insert a new element, search the tree to find the leaf node where the new element should be added. Insert the new element into that node with the following steps:

1. If the node contains fewer than the maximum allowed number of elements, then there is room for the new element. Insert the new element in the node, keeping the node's elements ordered.
2. Otherwise the node is full, evenly split it into two nodes so:
    1. A single median is chosen from among the leaf's elements and the new element.
    2. Values less than the median are put in the new left node and values greater than the median are put in the new right node, with the median acting as a separation value.
    3. The separation value is inserted in the node's parent, which may cause it to be split, and so on. If the node has no parent (i.e., the node was the root), create a new root above this node (increasing the height of the tree).

If the splitting goes all the way up to the root, it creates a new root with a single separator value and two children, which is why the lower bound on the size of internal nodes does not apply to the root. The maximum number of elements per node is $U-1$. When a node is split, one element moves to the parent, but one element is added. So, it must be possible to divide the maximum number $U-1$ of elements into two legal nodes. If this number is odd, then $U=2L$ and one of the new nodes contains $(U-2)/2 = L-1$ elements, and hence is a legal node, and the other contains one more element, and hence it is legal too. If $U-1$ is even, then $U=2L-1$, so there are $2L-2$ elements in the node. Half of this number is $L-1$, which is the minimum number of elements allowed per node.

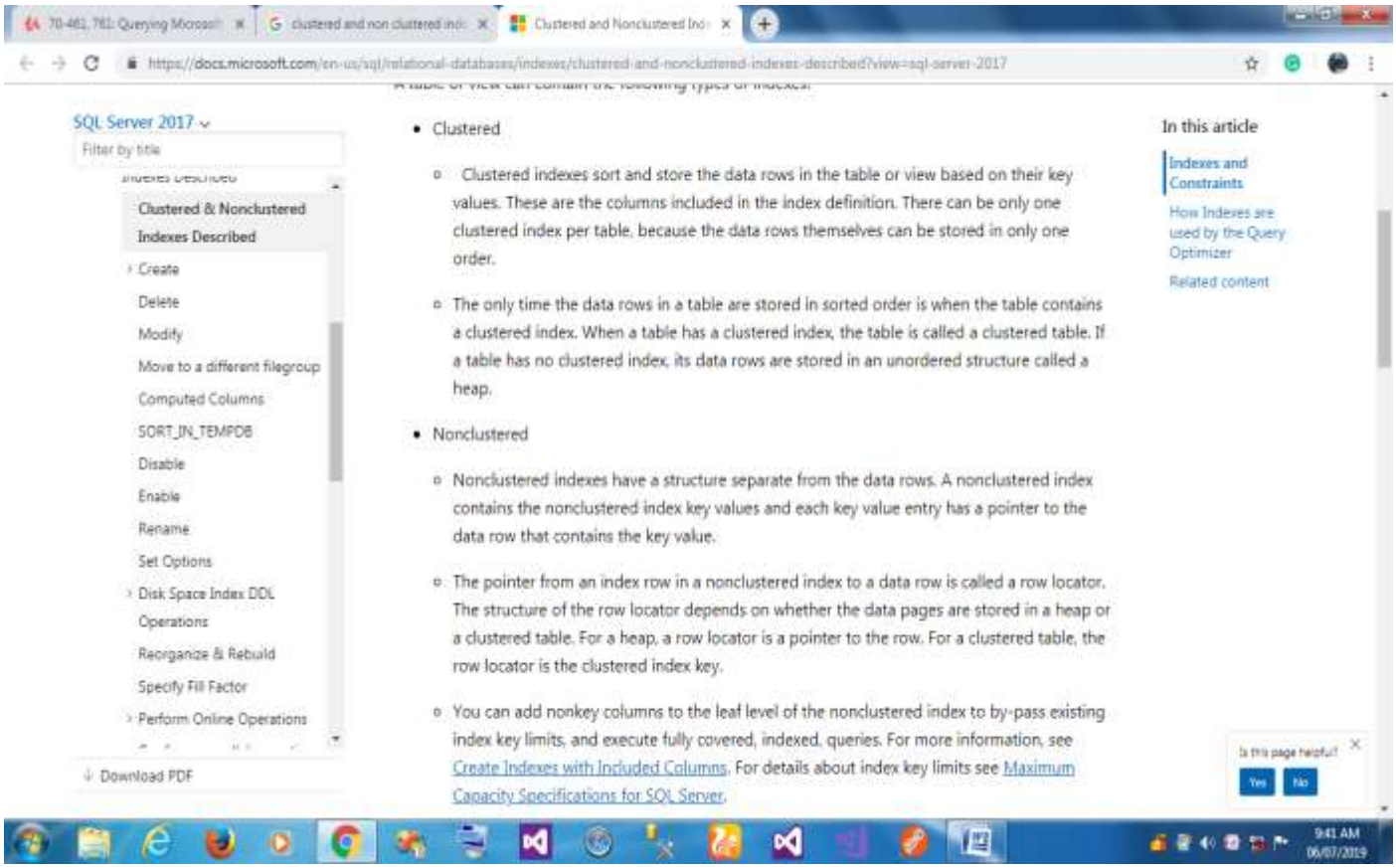
An improved algorithm[citation needed] supports a single pass down the tree from the root to the node where the insertion will take place, splitting any full nodes encountered on the way. This prevents the need to recall the parent nodes into memory, which may be expensive if the nodes are on secondary storage. However, to use this improved algorithm, we must be able to send one element to the parent and split the remaining $U-2$ elements into two legal nodes, without adding a new element. This requires $U = 2L$ rather than $U = 2L-1$, which accounts for why some textbooks impose this requirement in defining B-trees.

A B Tree insertion example with each iteration. The nodes of this B tree have at most 3 children (Knuth order 3).

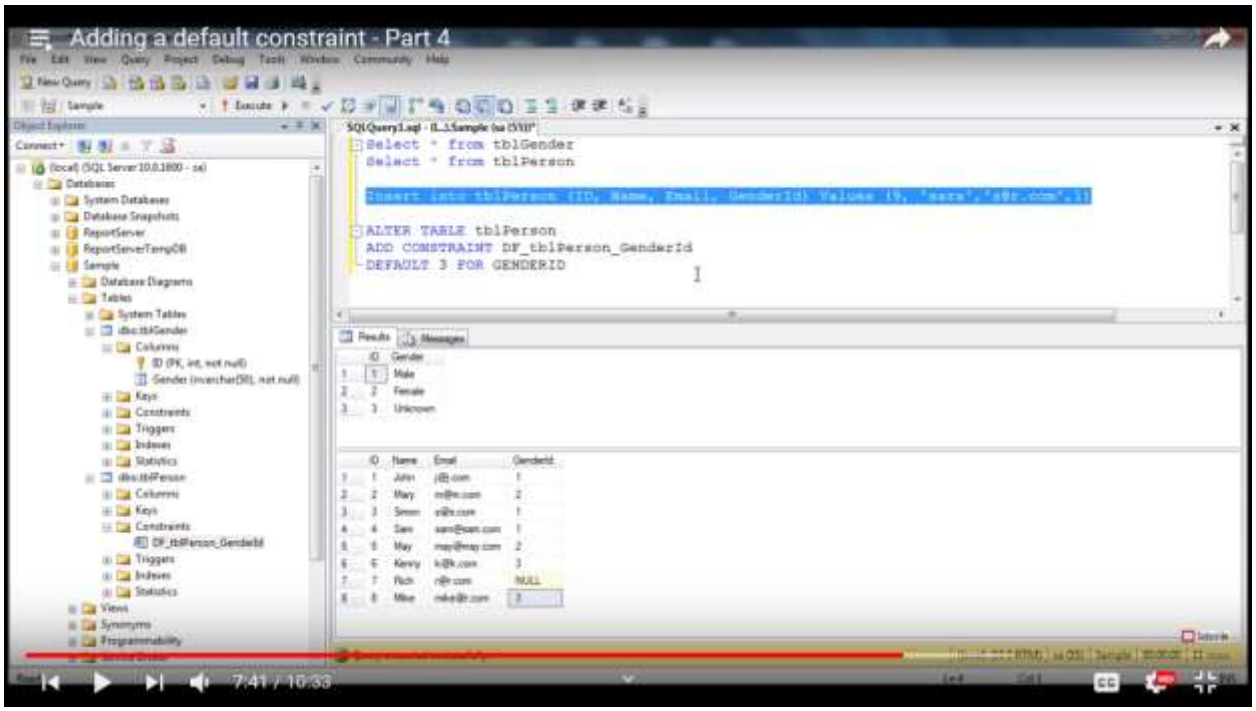


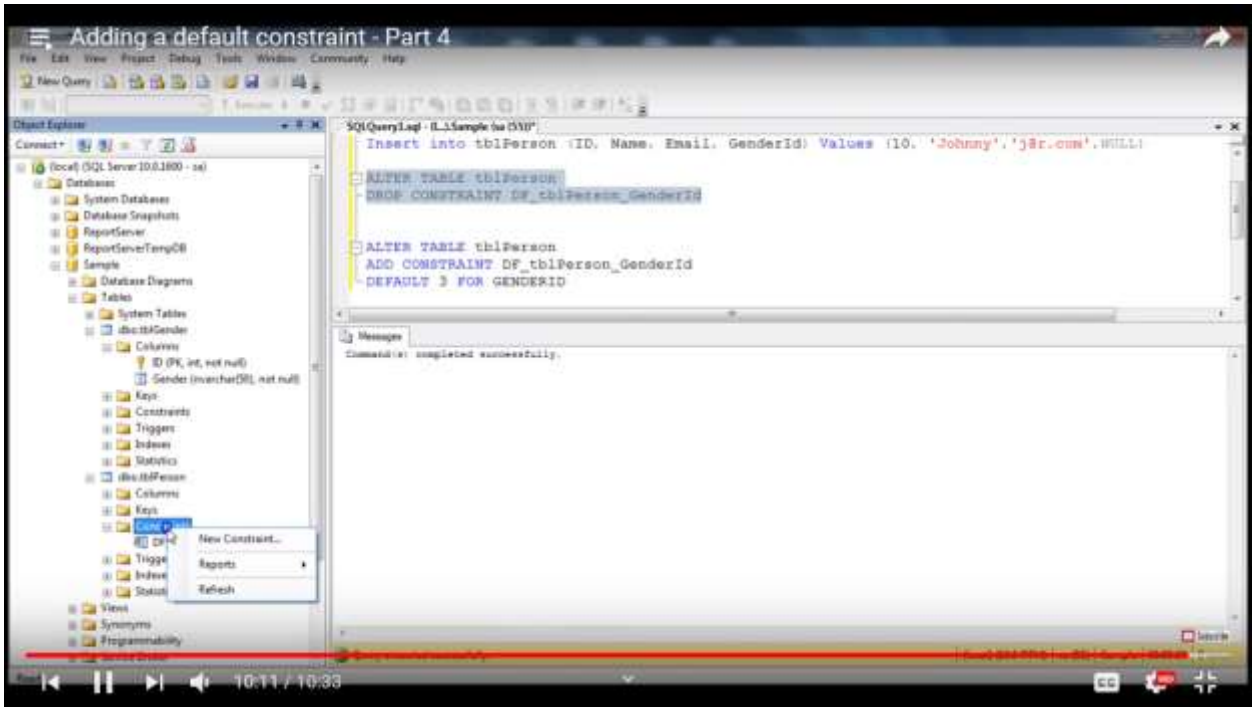

Adding a default constraint - Part 4
Default Constraint
tblPerson
ID Name Email GenderID
1 Jade j@j.com 2
2 Mary m@m.com 3
3 Martin ma@ma.com 3
4 Rob r@r.com NULL
5 May may@may.com 2
6 Kristy k@k.com NULL
tblGender
ID Gender
1 Male
2 Female
3 Unknown
A column default can be specified using Default constraint. The DEFAULT constraint is used to insert a default value into a column. The default value will be added to all new records, if no other value is specified, including NULL.
Altering an existing column to add a default constraint:
ALTER TABLE {TABLE_NAME}
ADD CONSTRAINT {CONSTRAINT_NAME}
DEFAULT {DEFAULT_VALUE} FOR {EXISTING_COLUMN_NAME}
Adding a new column, with default value, to an existing table:
ALTER TABLE {TABLE_NAME}
ADD {COLUMN_NAME} {DATA_TYPE} {NULL | NOT NULL}
CONSTRAINT {CONSTRAINT_NAME} DEFAULT {DEFAULT_VALUE}
Dropping a constraint:
ALTER TABLE {TABLE_NAME}
DROP CONSTRAINT {CONSTRAINT_NAME}
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:11 / 10:33

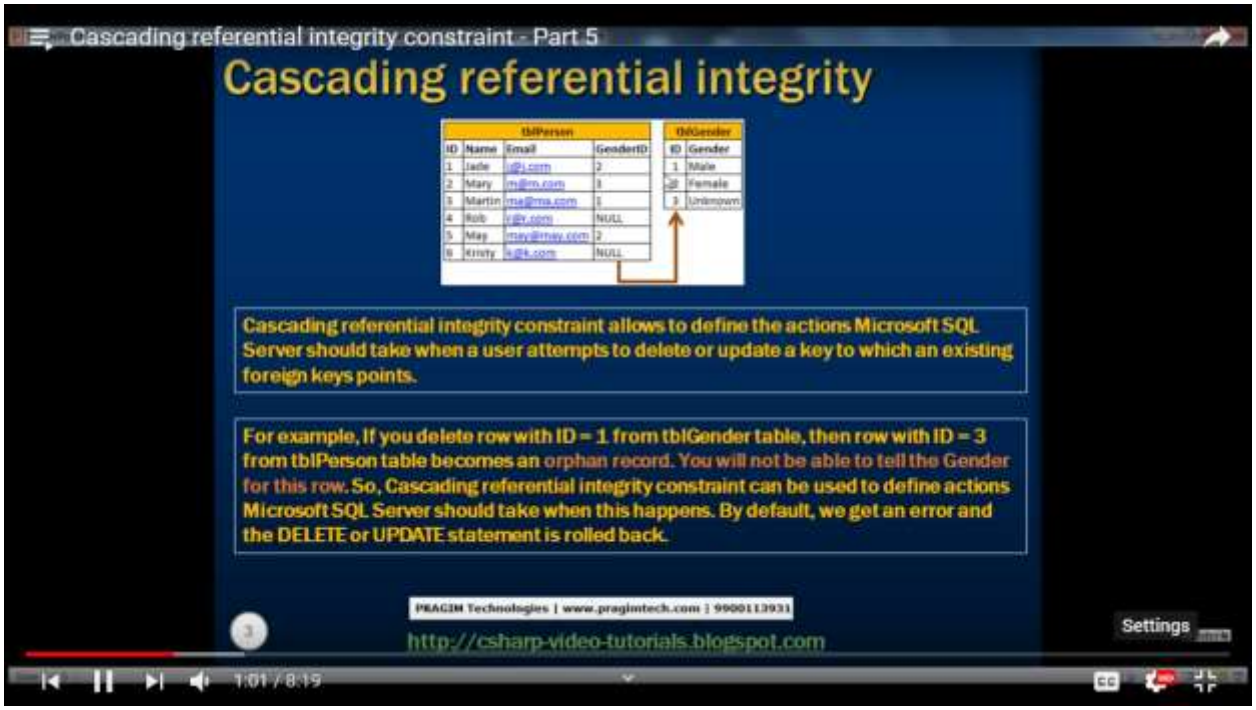
]

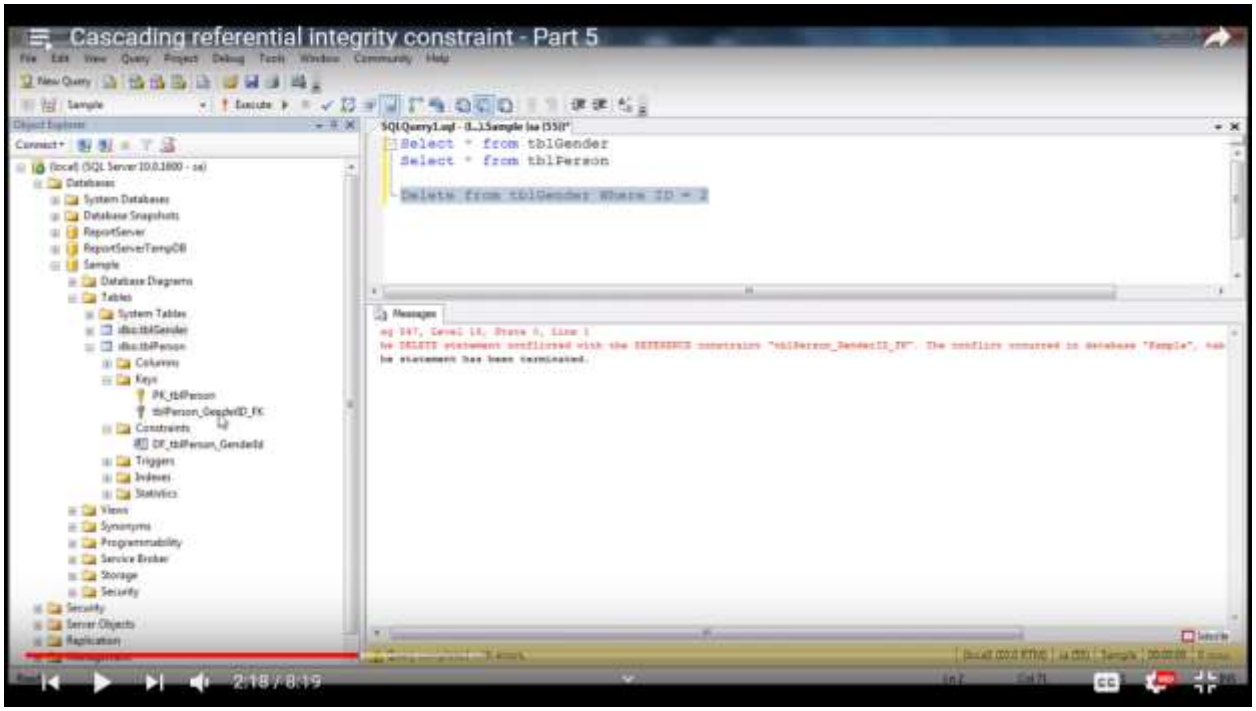

https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-2017
SQL Server 2017
Filter by title
Clustered & Nonclustered Indexes Described
Create
Delete
Modify
Move to a different filegroup
Computed Columns
SORT_IN_TEMPDB
Disable
Enable
Rename
Set Options
Disk Space Index DDL Operations
Reorganize & Rebuild
Specify Fill Factor
Perform Online Operations
Download PDF
Clustered
Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition. There can be only one clustered index per table, because the data rows themselves can be stored in only one order.
The only time the data rows in a table are stored in sorted order is when the table contains a clustered index. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.
Nonclustered
Nonclustered indexes have a structure separate from the data rows. A nonclustered index contains the nonclustered index key values and each key value entry has a pointer to the data row that contains the key value.
The pointer from an index row in a nonclustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.
You can add nonkey columns to the leaf level of the nonclustered index to by-pass existing index key limits, and execute fully covered, indexed, queries. For more information, see Create Indexes with Included Columns. For details about index key limits see Maximum Capacity Specifications for SQL Server.
In this article
Indexes and Constraints
How Indexes are used by the Query Optimizer
Related content
Is this page helpful?
Yes
No
9:41 AM
06/07/2019

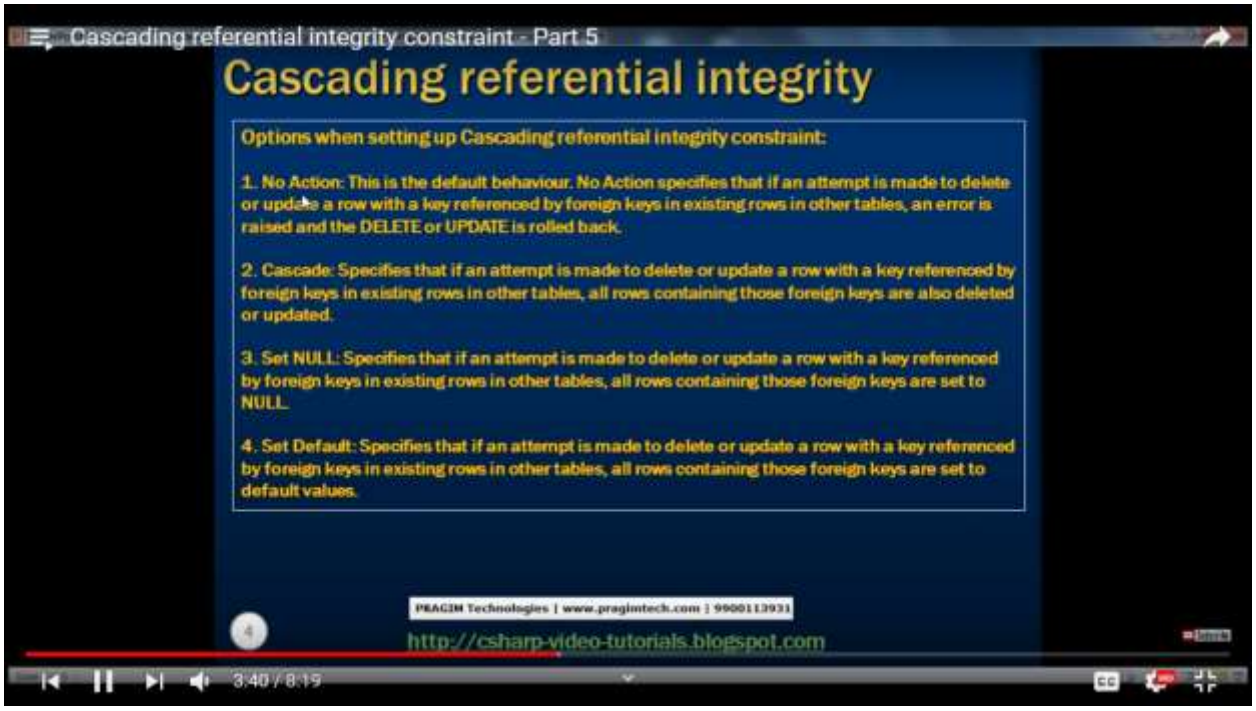

Adding a default constraint - Part 4
Default Constraint
tblPerson
ID Name Email GenderID
1 Jade j@j.com 2
2 Mary m@m.com 3
3 Martin ma@ma.com 3
4 Rob r@r.com NULL
5 May may@may.com 2
6 Kristy k@k.com NULL
tblGender
ID Gender
1 Male
2 Female
3 Unknown
A column default can be specified using Default constraint. The DEFAULT constraint is used to insert a default value into a column. The default value will be added to all new records, if no other value is specified, including NULL.
Altering an existing column to add a default constraint:
ALTER TABLE {TABLE_NAME}
ADD CONSTRAINT {CONSTRAINT_NAME}
DEFAULT {DEFAULT_VALUE} FOR {EXISTING_COLUMN_NAME}
Adding a new column, with default value, to an existing table:
ALTER TABLE {TABLE_NAME}
ADD {COLUMN_NAME} {DATA_TYPE} {NULL | NOT NULL}
CONSTRAINT {CONSTRAINT_NAME} DEFAULT {DEFAULT_VALUE}
Dropping a constraint:
ALTER TABLE {TABLE_NAME}
DROP CONSTRAINT {CONSTRAINT_NAME}
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:11 / 10:33

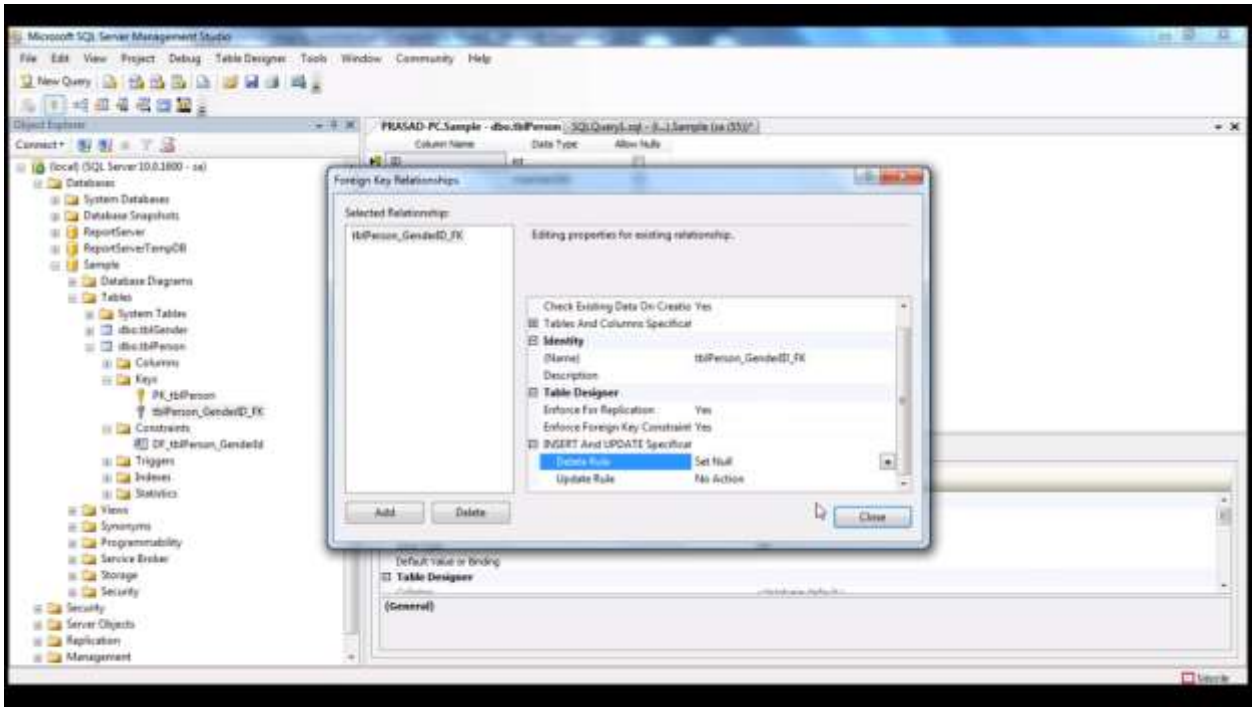

Adding a default constraint - Part 4
Select * from tblGender
Select * from tblPerson
Insert into tblPerson (ID, Name, Email, GenderId) Values (9, 'sara','s@r.com',1)
ALTER TABLE tblPerson
ADD CONSTRAINT DF_tblPerson_GenderId
DEFAULT 3 FOR GENDERID
7:41 / 10:33


Adding a default constraint - Part 4
Insert into tblPerson (ID, Name, Email, GenderId) Values (10, 'Johnny','j@r.com',NULL)
ALTER TABLE tblPerson
DROP CONSTRAINT DF_tblPerson_GenderId
ALTER TABLE tblPerson
ADD CONSTRAINT DF_tblPerson_GenderId
DEFAULT 3 FOR GENDERID
New Constraint...
Reports
Refresh
10:11 / 10:33

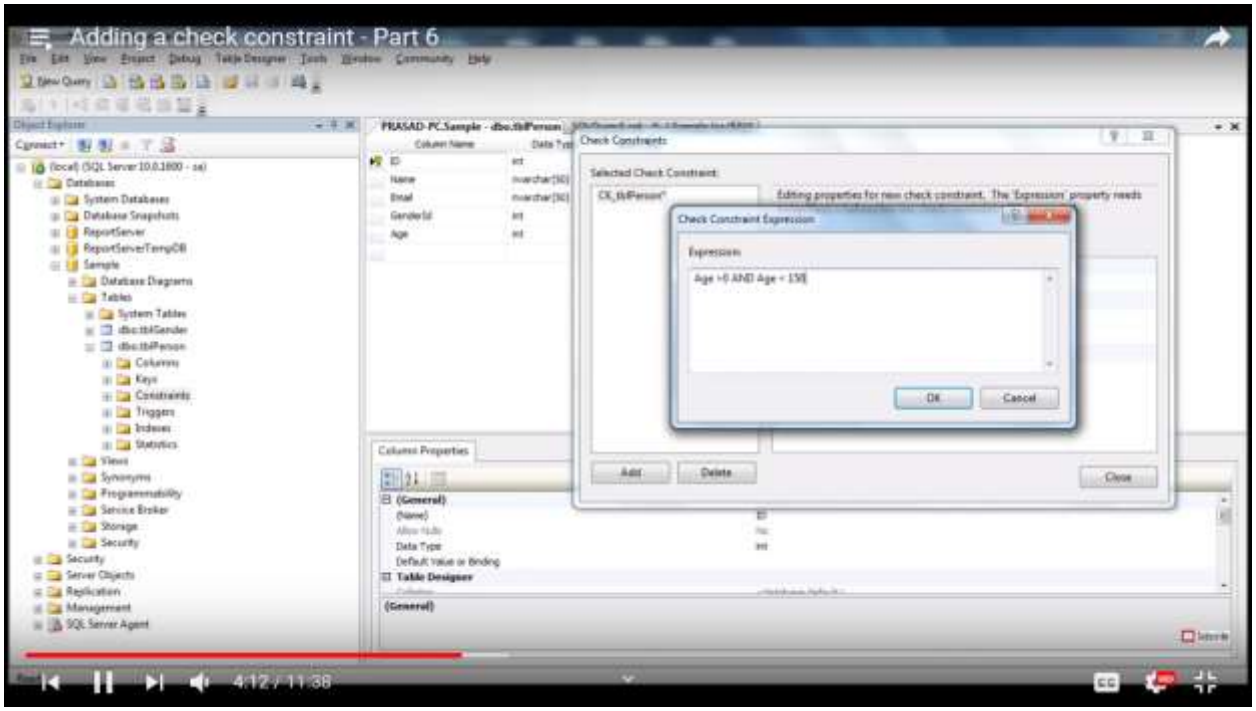

Cascading referential integrity constraint - Part 5
Cascading referential integrity
tblPerson
ID Name Email GenderID
1 Jade j@j.com 2
2 Mary m@m.com 3
3 Martin ma@ma.com 1
4 Rob r@r.com NULL
5 May may@may.com 2
6 Kristy k@k.com NULL
tblGender
ID Gender
1 Male
2 Female
3 Unknown
Cascading referential integrity constraint allows to define the actions Microsoft SQL Server should take when a user attempts to delete or update a key to which an existing foreign keys points.
For example, If you delete row with ID = 1 from tblGender table, then row with ID = 3 from tblPerson table becomes an orphan record. You will not be able to tell the Gender for this row. So, Cascading referential integrity constraint can be used to define actions Microsoft SQL Server should take when this happens. By default, we get an error and the DELETE or UPDATE statement is rolled back.
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
Settings
1:01 / 8:19

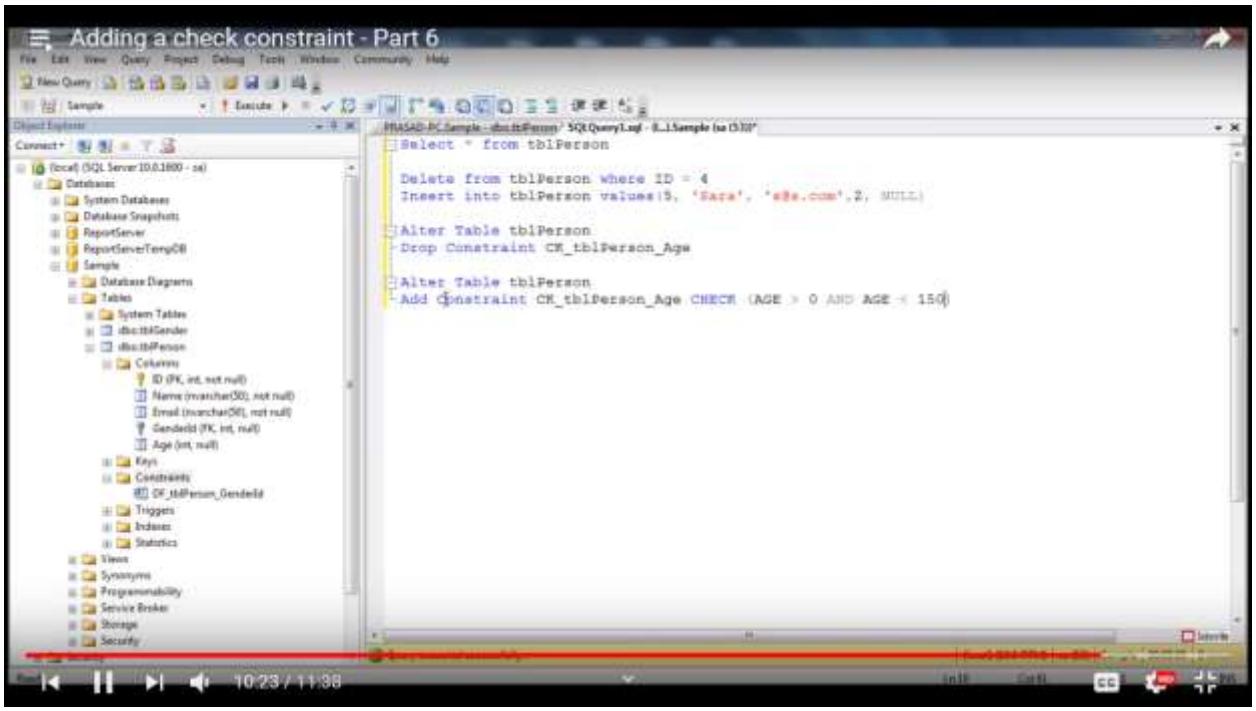

Cascading referential integrity constraint - Part 5
Select * from tblGender
Select * from tblPerson
Delete from tblGender Where ID = 2
2:18 / 8:19


Cascading referential integrity constraint - Part 5
Cascading referential integrity
Options when setting up Cascading referential integrity constraint:
1. No Action: This is the default behaviour. No Action specifies that if an attempt is made to delete or update a row with a key referenced by foreign keys in existing rows in other tables, an error is raised and the DELETE or UPDATE is rolled back.
2. Cascade: Specifies that if an attempt is made to delete or update a row with a key referenced by foreign keys in existing rows in other tables, all rows containing those foreign keys are also deleted or updated.
3. Set NULL: Specifies that if an attempt is made to delete or update a row with a key referenced by foreign keys in existing rows in other tables, all rows containing those foreign keys are set to NULL.
4. Set Default: Specifies that if an attempt is made to delete or update a row with a key referenced by foreign keys in existing rows in other tables, all rows containing those foreign keys are set to default values.
PRAGIM Technologies | www.pragimtech.com | 9900113931
4
http://csharp-video-tutorials.blogspot.com
3:40 / 8:19

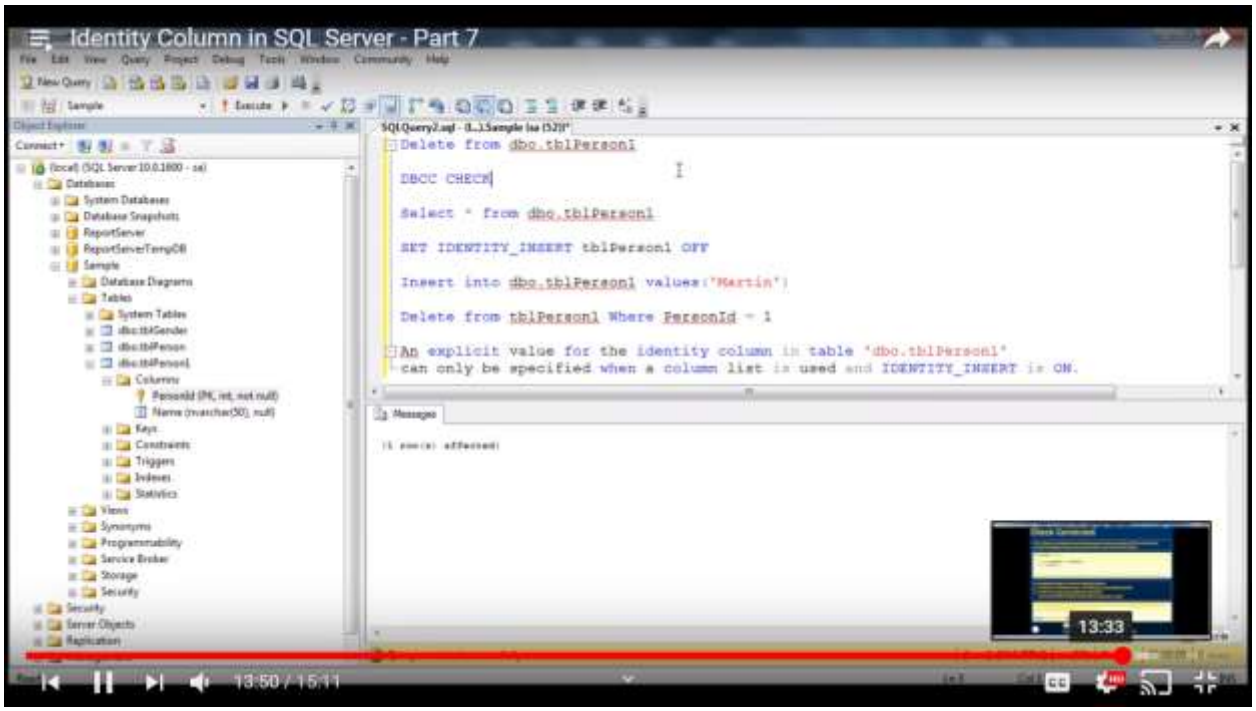

Microsoft SQL Server Management Studio
Foreign Key Relationships
Selected Relationship:
tblPerson_GenderID_FK
Editing properties for existing relationship.
Identity
Table Designer
Add
Delete
Close

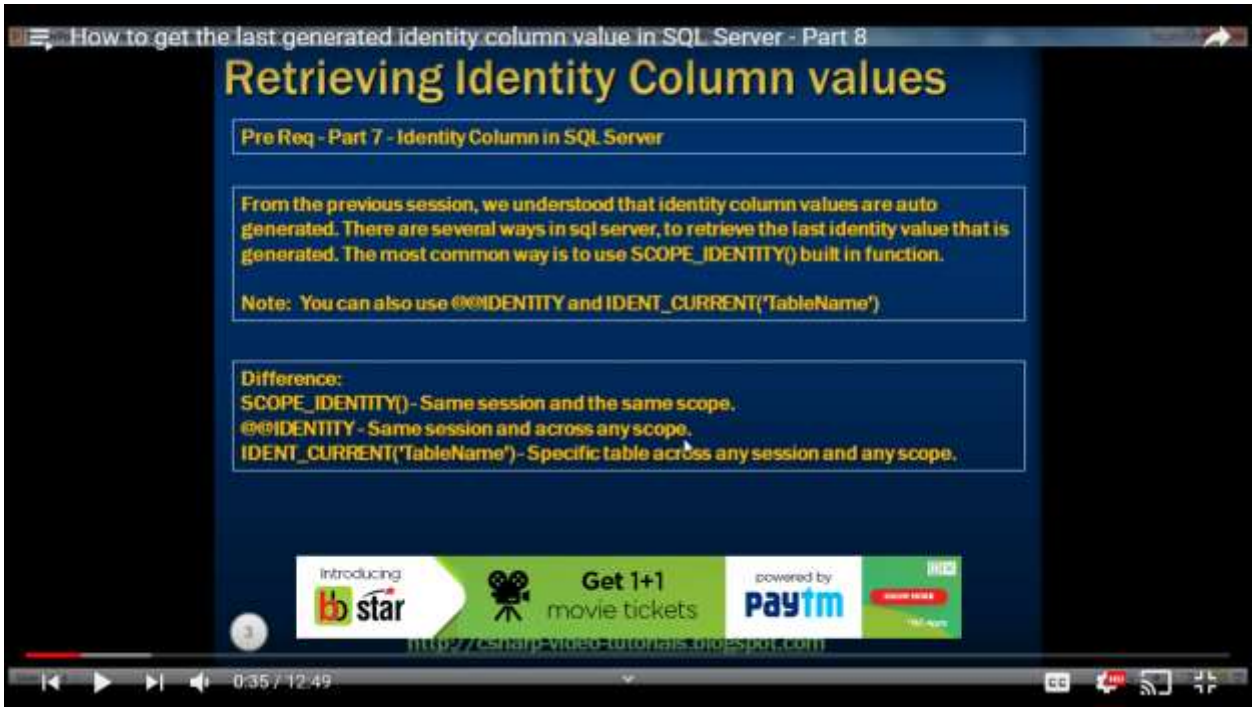

Adding a check constraint - Part 6
Check Constraint
CHECK constraint is used to limit the range of the values, that can be entered for a column.
The general formula for adding check constraint in SQL Server:
ALTER TABLE {TABLE_NAME}
ADD CONSTRAINT {CONSTRAINT_NAME} CHECK (BOOLEAN_EXPRESSION)
If the BOOLEAN_EXPRESSION returns true, then the CHECK constraint allows the value, otherwise it doesn't. Since, AGE is a nullable column, it's possible to pass null for this column, when inserting a row. When you pass NULL for the AGE column, the boolean expression evaluates to UNKNOWN, and allows the value.
To drop the CHECK constraint:
ALTER TABLE tblPerson
DROP CONSTRAINT CK_tblPerson_Age
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:12 / 11:38

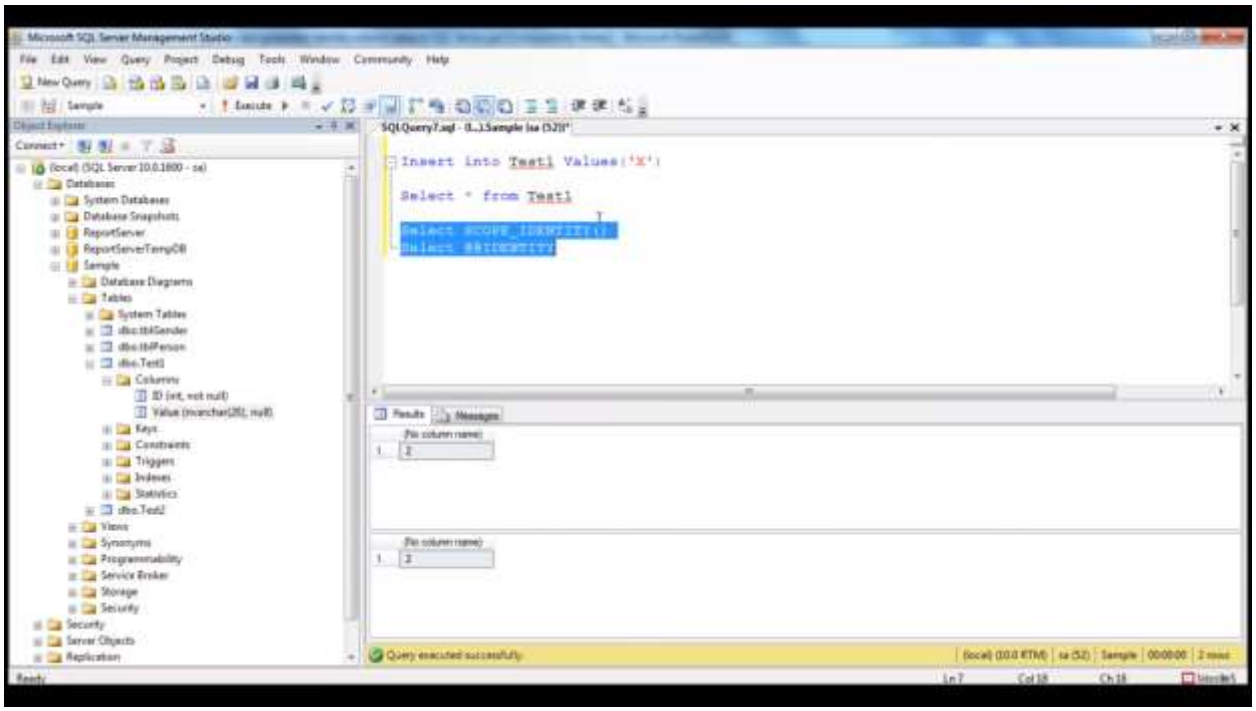

Adding a check constraint - Part 6
Check Constraints
Selected Check Constraint:
Check Constraint Expression
Expression:
Age >0 AND Age < 150
OK
Cancel
Add
Delete
Close
4:12 / 11:38

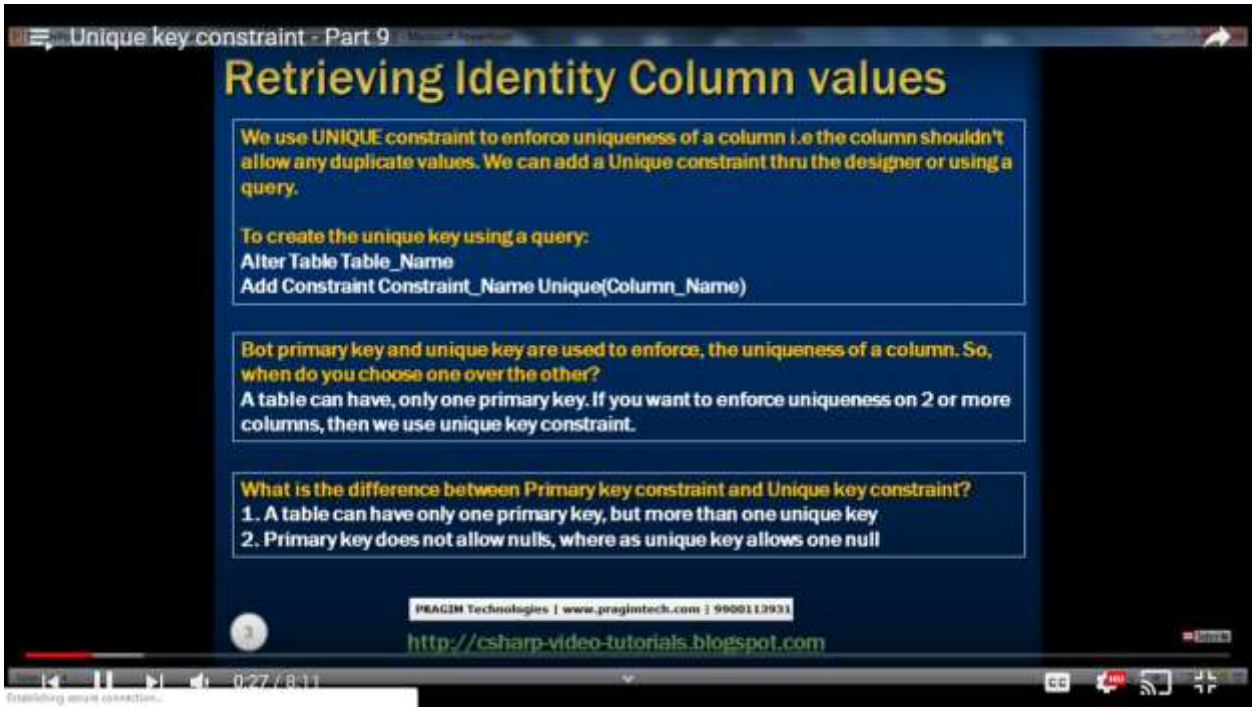

Adding a check constraint - Part 6
Select * from tblPerson
Delete from tblPerson where ID = 4
Insert into tblPerson values(5, 'Sara', 'sara.com',2, NULL)
Alter Table tblPerson
Drop Constraint CK_tblPerson_Age
Alter Table tblPerson
Add Constraint CK_tblPerson_Age CHECK (AGE > 0 AND AGE < 150)
10:23 / 11:38

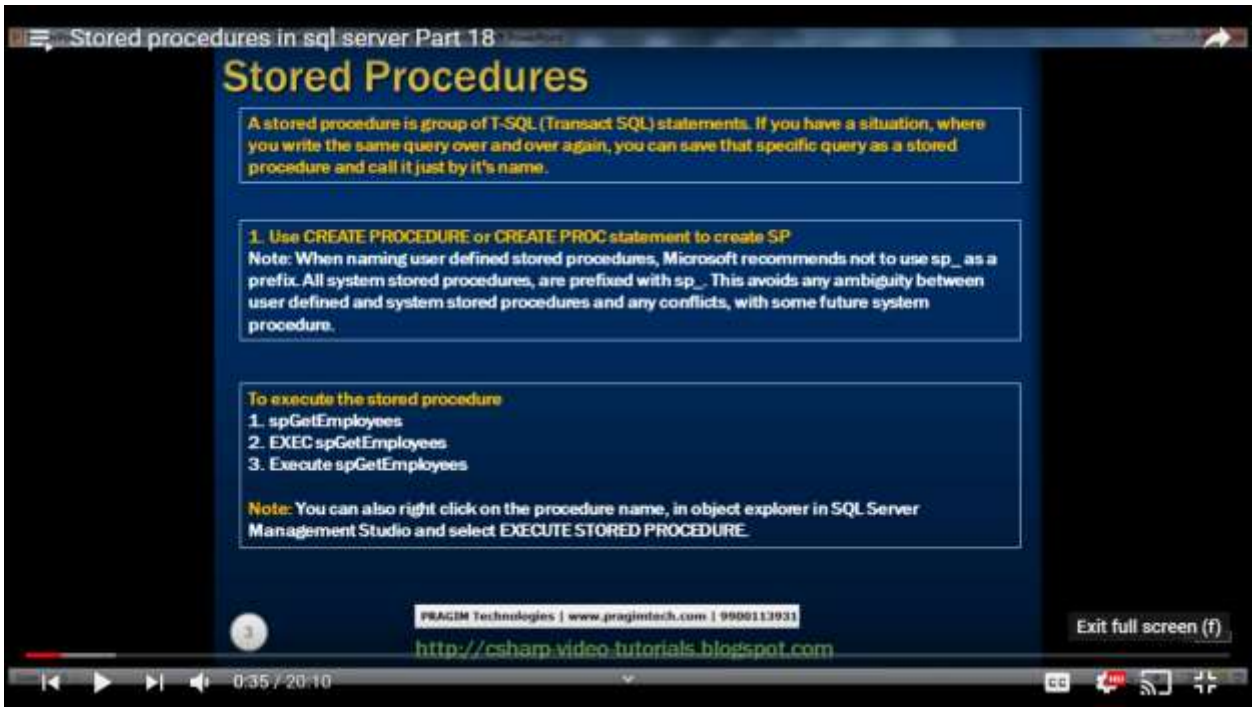

Identity Column in SQL Server - Part 7
Check Constraint
If a column is marked as an identity column, then the values for this column are automatically generated, when you insert a new row into the table.
Create Table tblPerson
(
PersonId int Identity(1,1) Primary Key
Name nvarchar(20)
)
Note: Seed and Increment values are optional. If you don't specify the identity and seed they both default to 1.
To explicitly supply a value for identity column
1. First turn on identity insert - SET Identity_Insert tblPerson ON
2. In the insert query specify the column list
Insert into tblPerson(PersonId, Name) values(2, 'John')
If you have deleted all the rows in a table, and you want to reset the identity column value, use DBCC CHECKIDENT command.
DBCC CHECKIDENT('tblPerson', RESEED, 0)
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:21 / 15:11

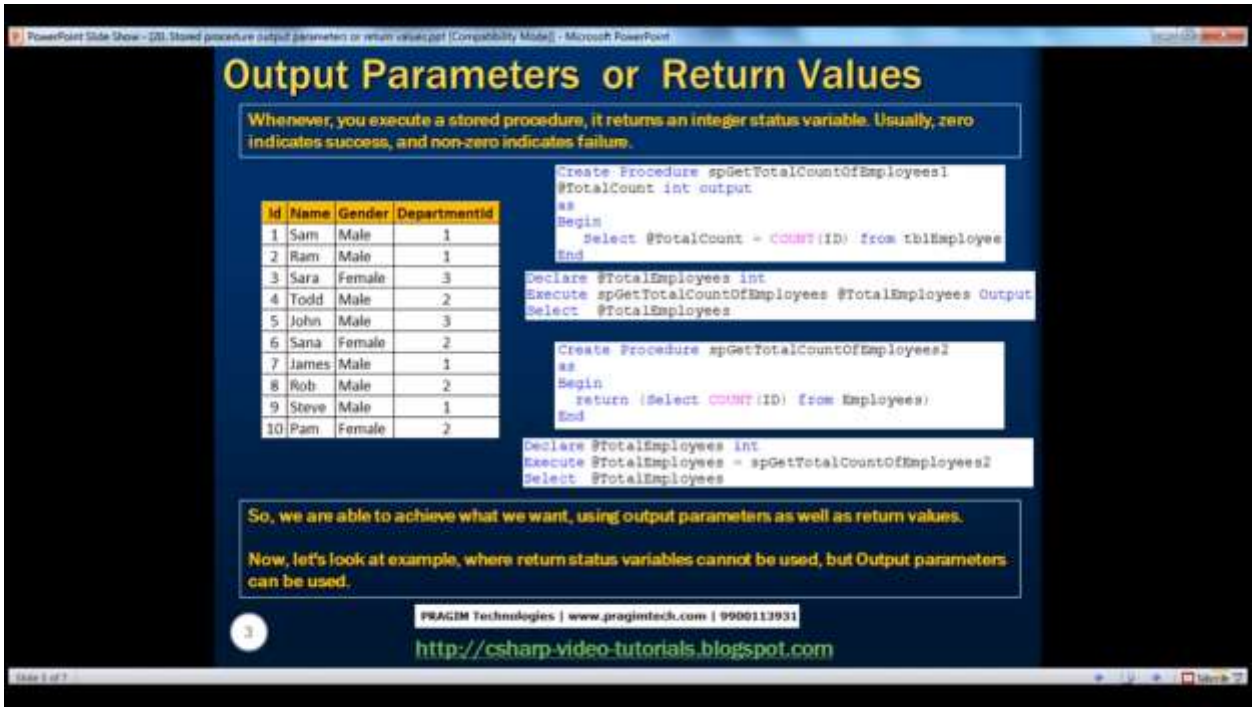

Identity Column in SQL Server - Part 7
Delete from dbo.tblPerson1
DBCC CHECK
Select * from dbo.tblPerson1
SET IDENTITY_INSERT tblPerson1 OFF
Insert into dbo.tblPerson1 values('Martin')
Delete from tblPerson1 Where PersonId = 1
An explicit value for the identity column in table 'dbo.tblPerson1' can only be specified when a column list is used and IDENTITY_INSERT is ON.
13:33
13:50 / 15:11

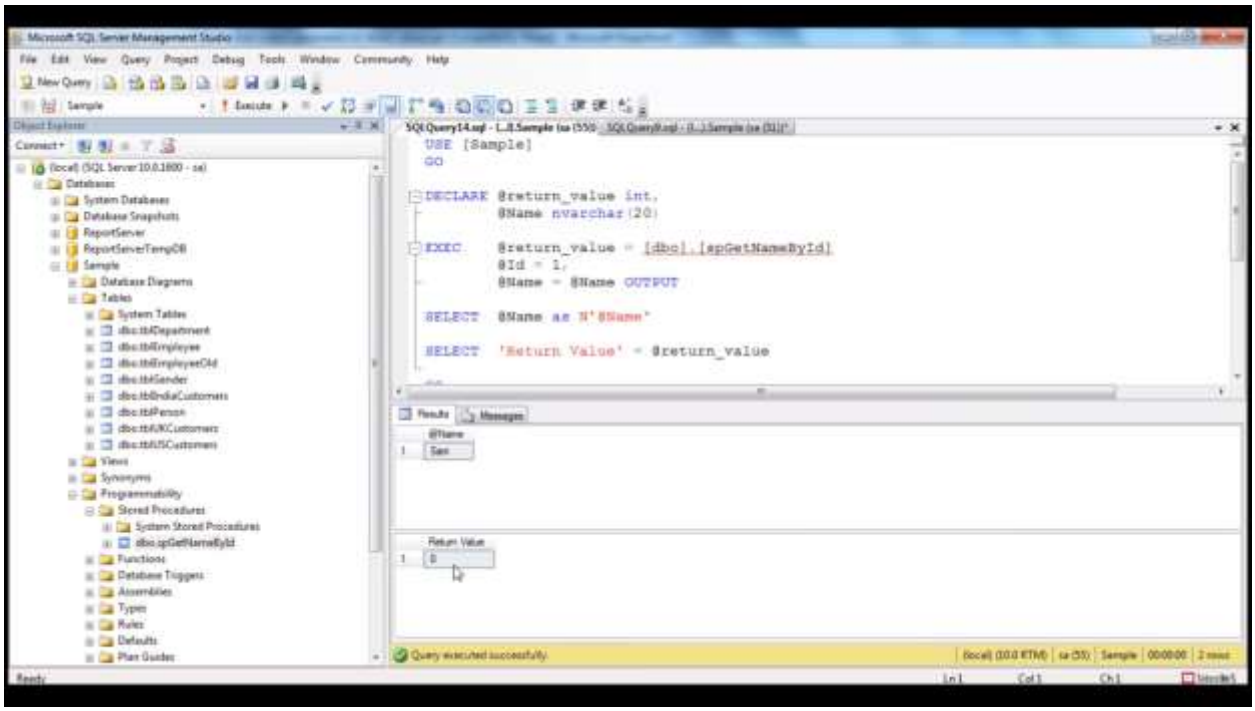

How to get the last generated identity column value in SQL Server - Part 8
Retrieving Identity Column values
Pre Req - Part 7 - Identity Column in SQL Server
From the previous session, we understood that identity column values are auto generated. There are several ways in sql server, to retrieve the last identity value that is generated. The most common way is to use SCOPE_IDENTITY() built in function.
Note: You can also use @@IDENTITY and IDENT_CURRENT('TableName')
Difference:
SCOPE_IDENTITY() - Same session and the same scope.
@@IDENTITY - Same session and across any scope.
IDENT_CURRENT('TableName') - Specific table across any session and any scope.
Introducing
bb star
Get 1+1
movie tickets
powered by
paytm
0:35 / 12:49

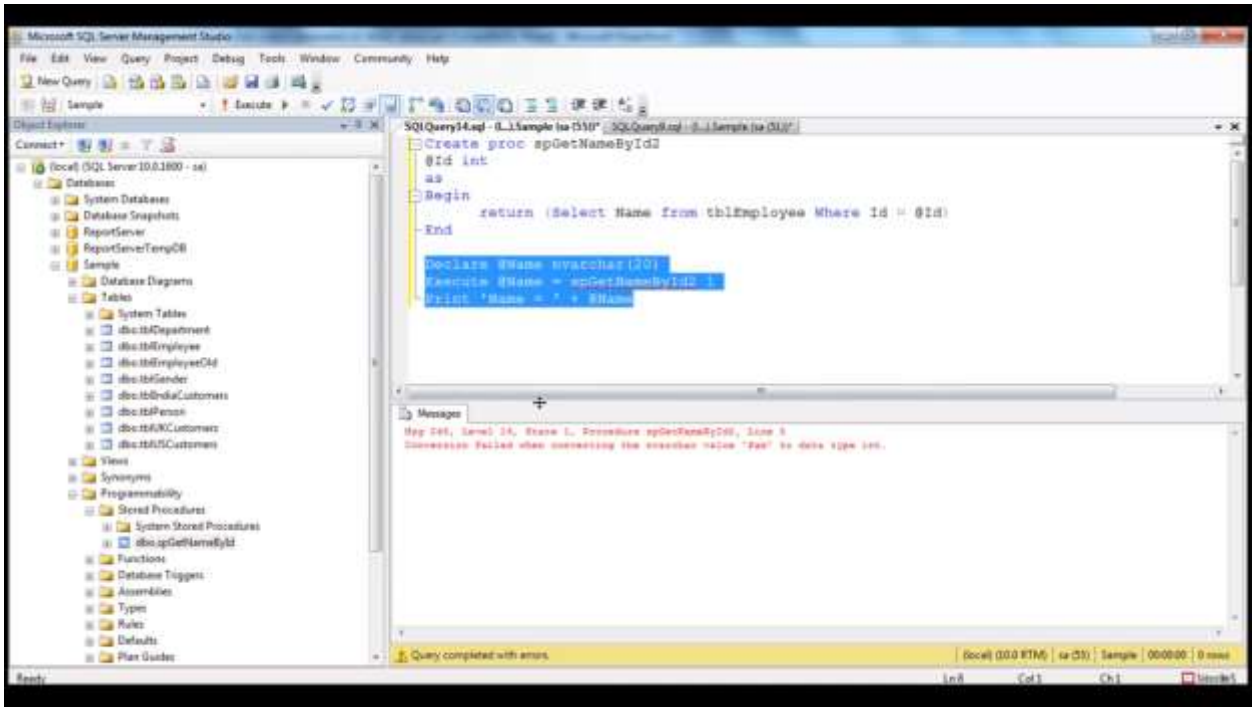

Insert into Test1 Values('X')
Select * from Test1
Select SCOPE_IDENTITY()
Select @@IDENTITY

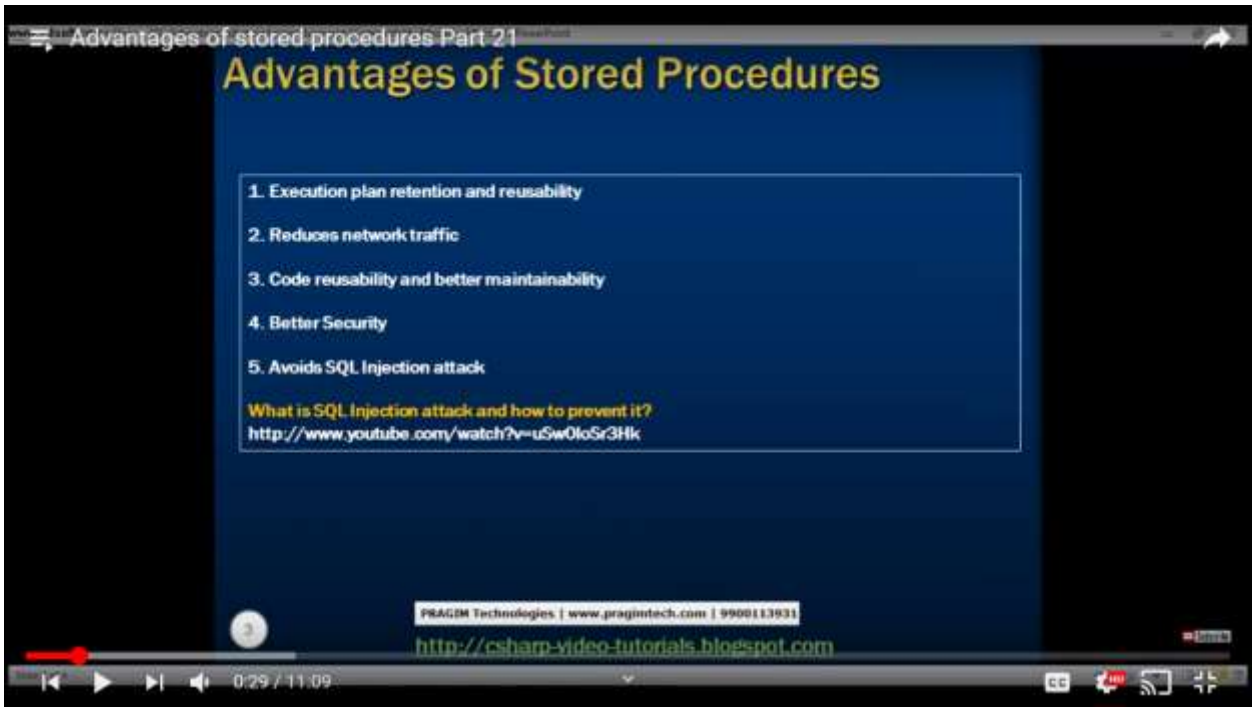

Unique key constraint - Part 9
Retrieving Identity Column values
We use UNIQUE constraint to enforce uniqueness of a column i.e the column shouldn't allow any duplicate values. We can add a Unique constraint thru the designer or using a query.
To create the unique key using a query:
Alter Table Table_Name
Add Constraint Constraint_Name Unique(Column_Name)
Bot primary key and unique key are used to enforce, the uniqueness of a column. So, when do you choose one over the other?
A table can have, only one primary key. If you want to enforce uniqueness on 2 or more columns, then we use unique key constraint.
What is the difference between Primary key constraint and Unique key constraint?
1. A table can have only one primary key, but more than one unique key
2. Primary key does not allow nulls, where as unique key allows one null
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:27 / 8:11


Stored procedures in sql server Part 18
Stored Procedures
A stored procedure is group of T-SQL (Transact SQL) statements. If you have a situation, where you write the same query over and over again, you can save that specific query as a stored procedure and call it just by it's name.
1. Use CREATE PROCEDURE or CREATE PROC statement to create SP
Note: When naming user defined stored procedures, Microsoft recommends not to use sp_ as a prefix. All system stored procedures, are prefixed with sp_. This avoids any ambiguity between user defined and system stored procedures and any conflicts, with some future system procedure.
To execute the stored procedure
1. spGetEmployees
2. EXEC spGetEmployees
3. Execute spGetEmployees
Note: You can also right click on the procedure name, in object explorer in SQL Server Management Studio and select EXECUTE STORED PROCEDURE.
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
Exit full screen (f)
0:35 / 20:10


Output Parameters or Return Values
Whenever, you execute a stored procedure, it returns an integer status variable. Usually, zero indicates success, and non-zero indicates failure.
Id Name Gender DepartmentId
1 Sam Male 1
2 Ram Male 1
3 Sara Female 3
4 Todd Male 2
5 John Male 3
6 Sana Female 2
7 James Male 1
8 Rob Male 2
9 Steve Male 1
10 Pam Female 2
Create Procedure spGetTotalCountOfEmployees1
@TotalCount int output
as
Begin
Select @TotalCount = COUNT(ID) from tblEmployee
End
Declare @TotalEmployees int
Execute spGetTotalCountOfEmployees @TotalEmployees Output
Select @TotalEmployees
Create Procedure spGetTotalCountOfEmployees2
as
Begin
return (Select COUNT(ID) from Employees)
End
Declare @TotalEmployees int
Execute @TotalEmployees = spGetTotalCountOfEmployees2
Select @TotalEmployees
So, we are able to achieve what we want, using output parameters as well as return values.
Now, let's look at example, where return status variables cannot be used, but Output parameters can be used.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

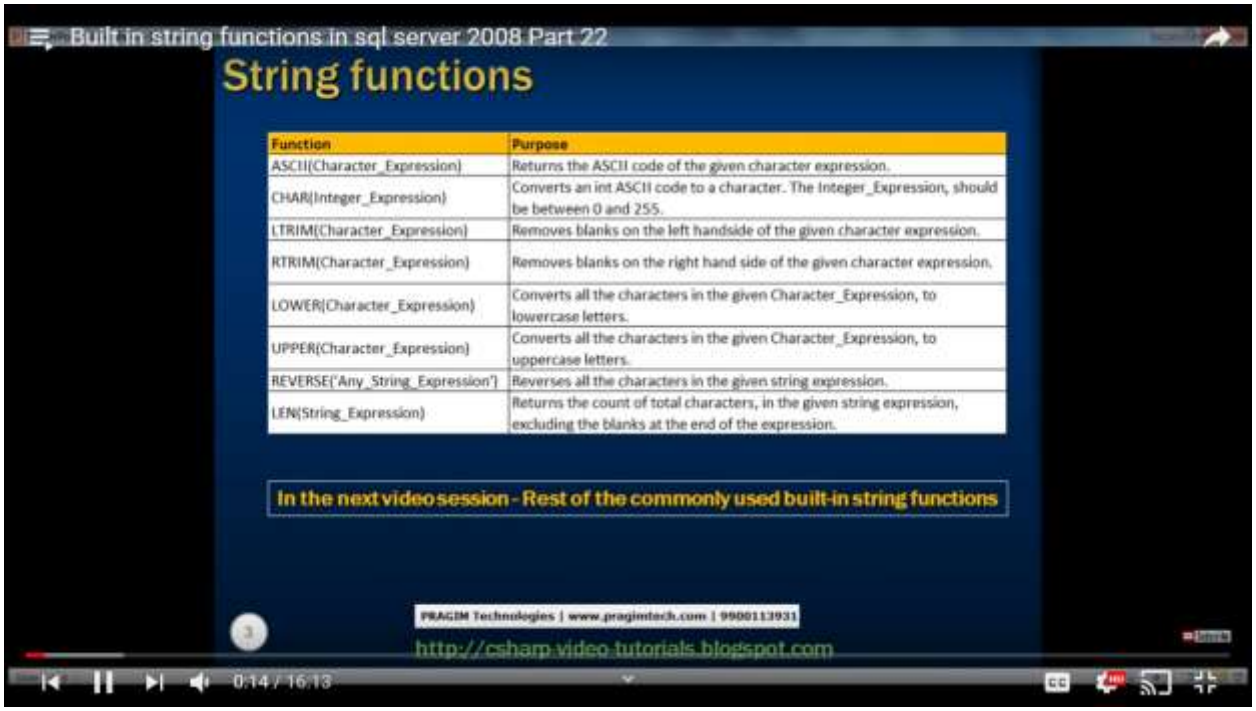

USE [Sample]
GO
DECLARE @return_value int,
@Name nvarchar(20)
EXEC @return_value = [dbo].[spGetNameById]
@Id = 1,
@Name = @Name OUTPUT
SELECT @Name as N'@Name'
SELECT 'Return Value' = @return_value
Results
Messages
@Name
1 Sam
Return Value
1 0
Query executed successfully.

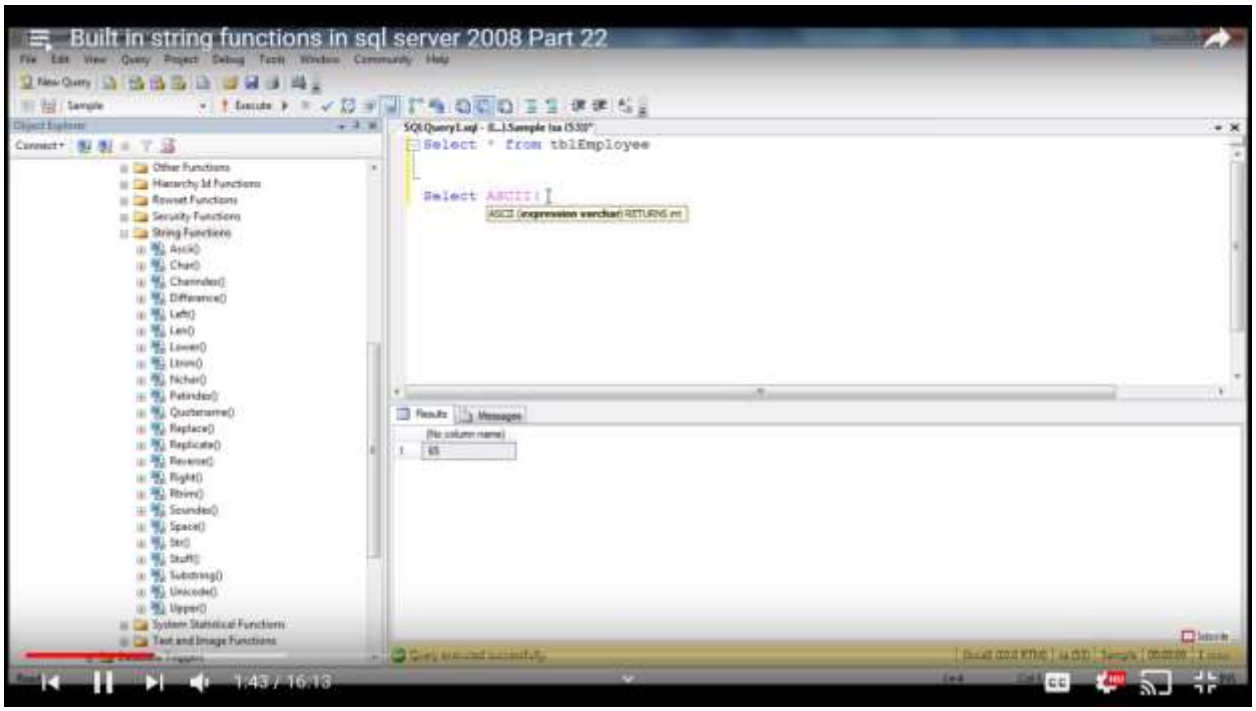


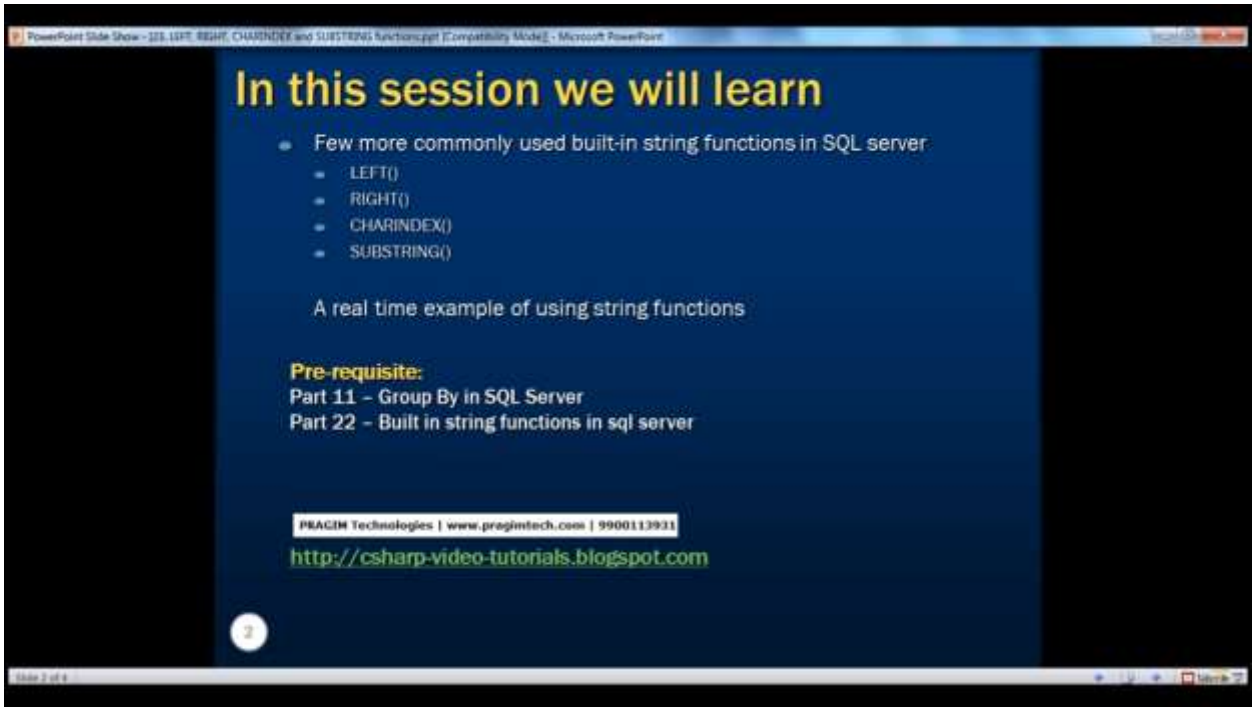


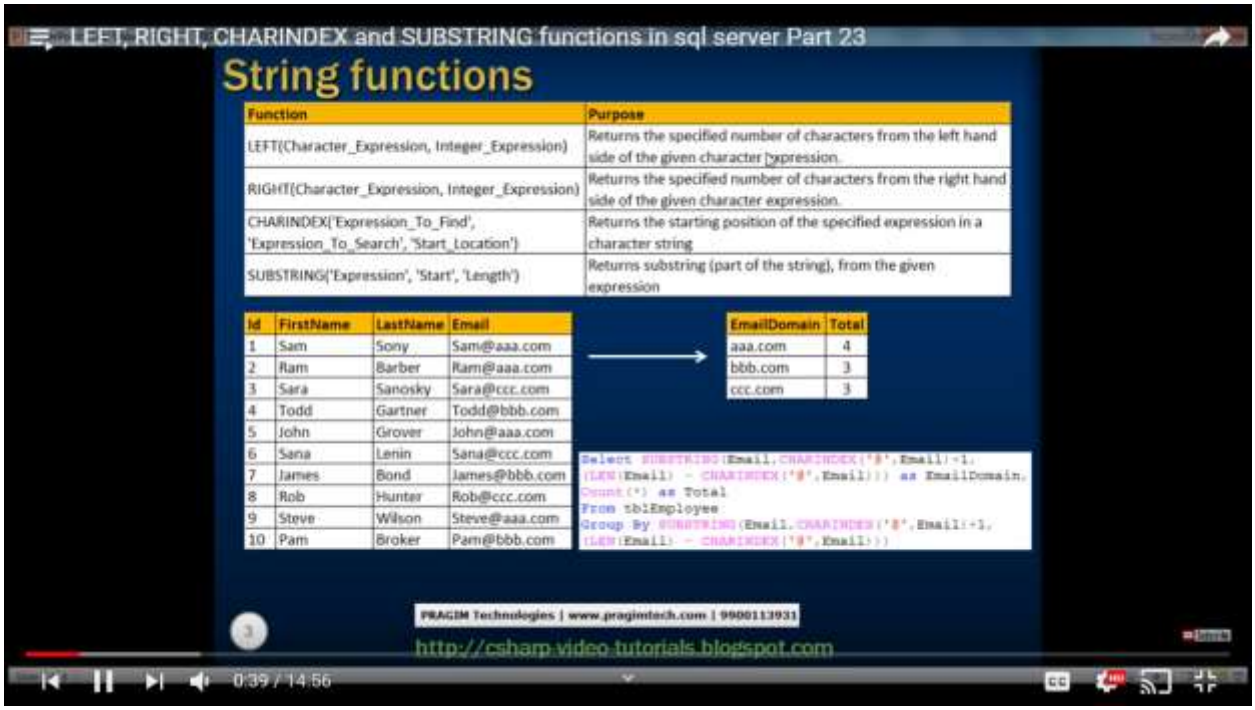
If adhoc query use execution plan but change in parameter it will not use same execution plan

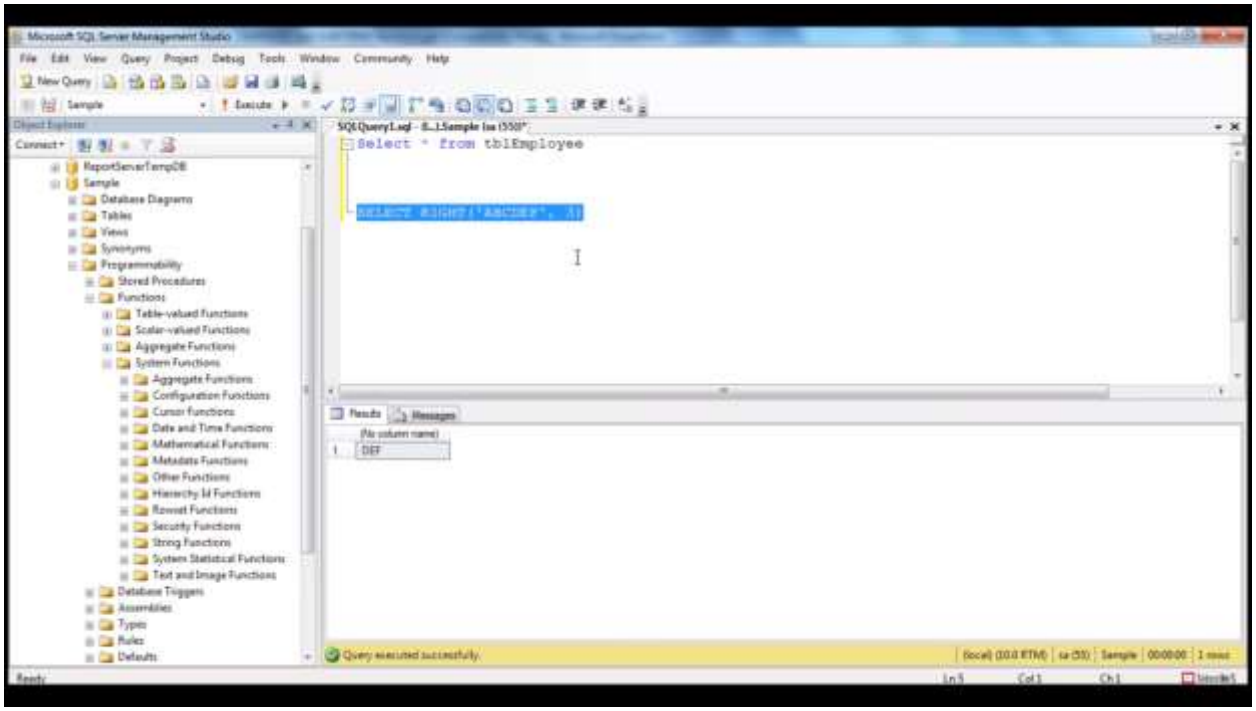
But store procedures use same execution plan again and again

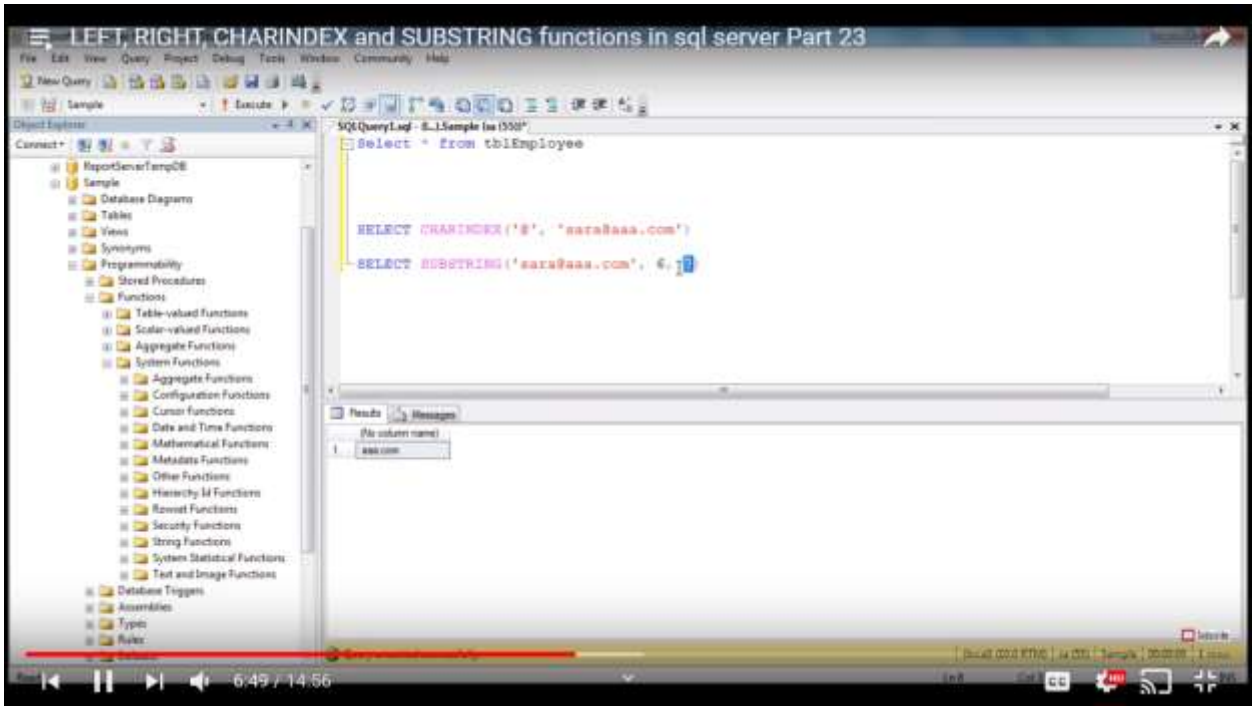

Built in string functions in sql server 2008 Part 22
String functions
Function
Purpose
ASCII(Character_Expression)
Returns the ASCII code of the given character expression.
CHAR(Integer_Expression)
Converts an int ASCII code to a character. The Integer_Expression, should be between 0 and 255.
LTRIM(Character_Expression)
Removes blanks on the left handside of the given character expression.
RTRIM(Character_Expression)
Removes blanks on the right hand side of the given character expression.
LOWER(Character_Expression)
Converts all the characters in the given Character_Expression, to lowercase letters.
UPPER(Character_Expression)
Converts all the characters in the given Character_Expression, to uppercase letters.
REVERSE('Any_String_Expression')
Reverses all the characters in the given string expression.
LEN(String_Expression)
Returns the count of total characters, in the given string expression, excluding the blanks at the end of the expression.
In the next video session - Rest of the commonly used built-in string functions
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:14 / 16:13

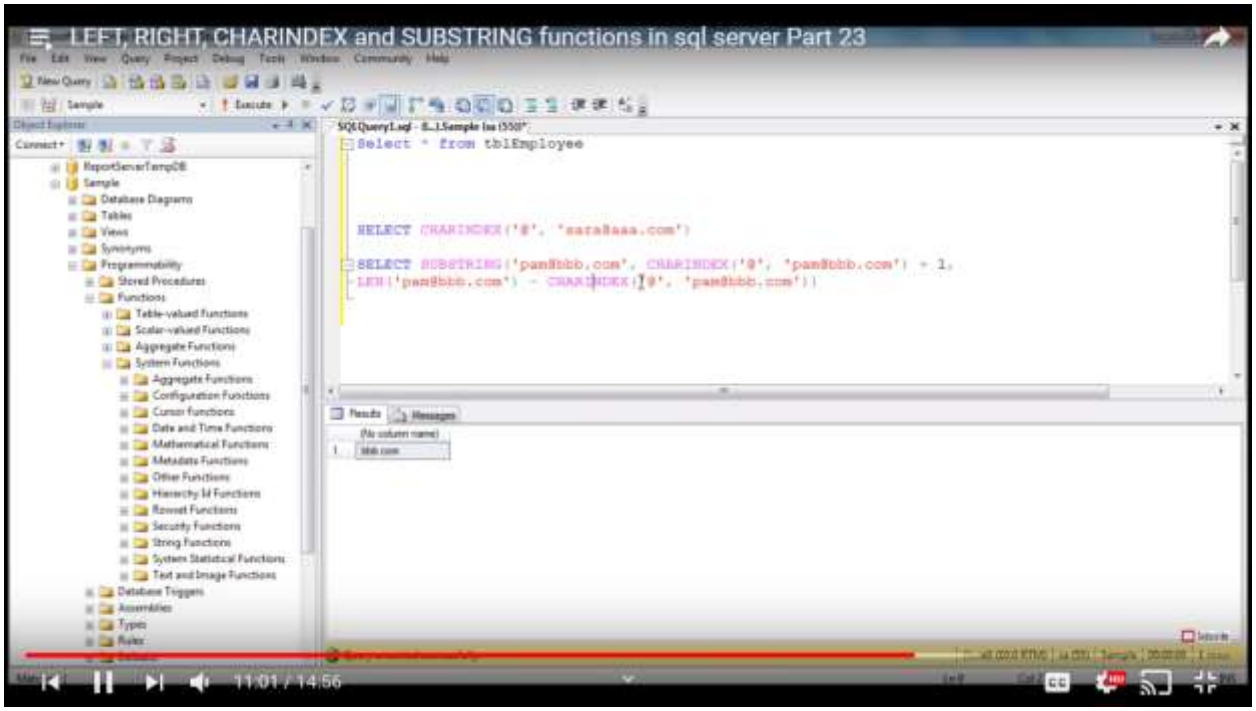

Built in string functions in sql server 2008 Part 22
Select * from tblEmployee
Select ASCII('
1:43 / 16:13

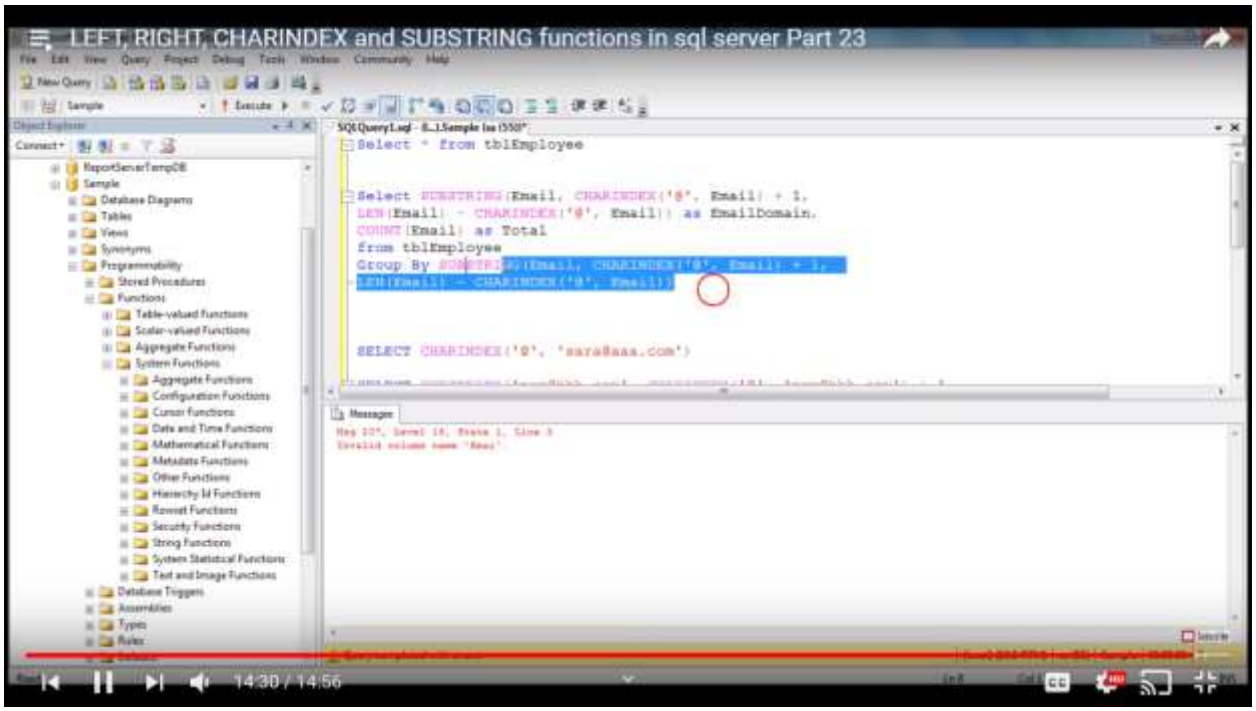

In this session we will learn
Few more commonly used built-in string functions in SQL server
LEFT()
RIGHT()
CHARINDEX()
SUBSTRING()
A real time example of using string functions
Pre-requisite:
Part 11 – Group By in SQL Server
Part 22 – Built in string functions in sql server
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
2

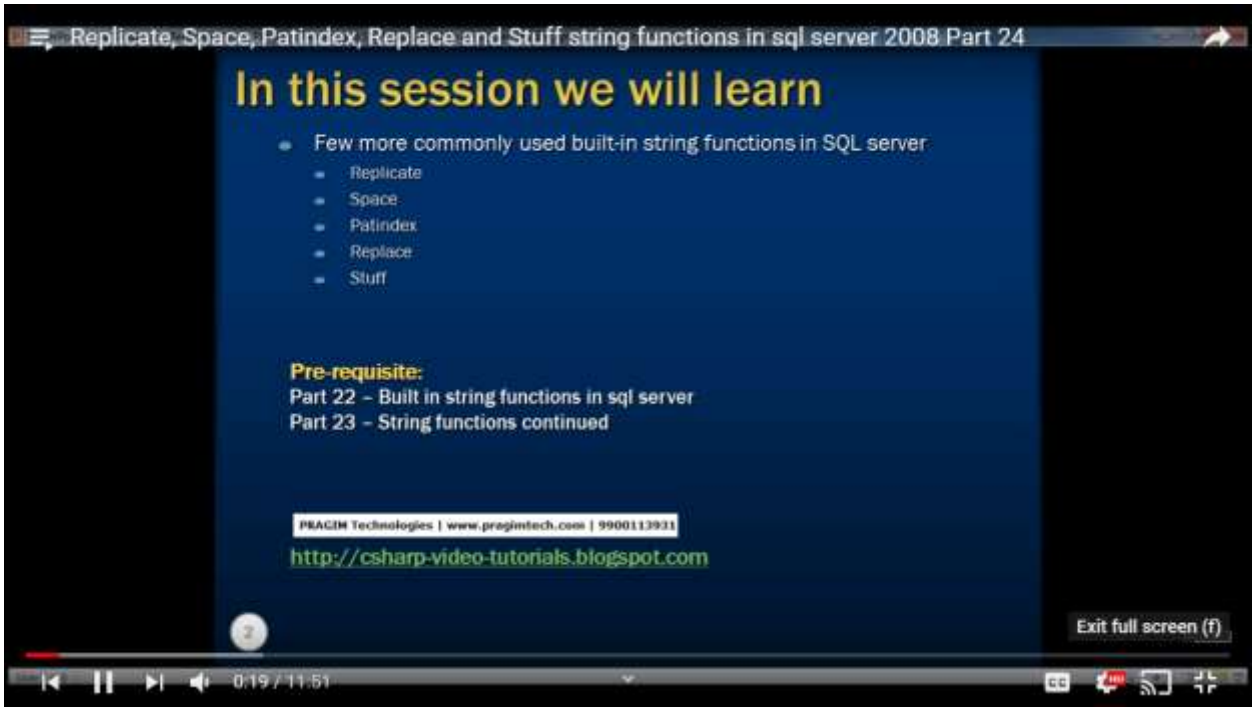

LEFT, RIGHT, CHARINDEX and SUBSTRING functions in sql server Part 23
String functions
Function
Purpose
LEFT(Character_Expression, Integer_Expression)
Returns the specified number of characters from the left hand side of the given character expression.
RIGHT(Character_Expression, Integer_Expression)
Returns the specified number of characters from the right hand side of the given character expression.
CHARINDEX('Expression_To_Find', 'Expression_To_Search', 'Start_Location')
Returns the starting position of the specified expression in a character string
SUBSTRING('Expression', 'Start', 'Length')
Returns substring (part of the string), from the given expression
Id
FirstName
LastName
Email
1 Sam Sony Sam@aaa.com
2 Ram Barber Ram@aaa.com
3 Sara Sanosky Sara@ccc.com
4 Todd Gartner Todd@bbb.com
5 John Grover John@aaa.com
6 Sana Lenin Sana@ccc.com
7 James Bond James@bbb.com
8 Rob Hunter Rob@ccc.com
9 Steve Wilson Steve@aaa.com
10 Pam Broker Pam@bbb.com
EmailDomain
Total
aaa.com 4
bbb.com 3
ccc.com 3
Select SUBSTRING(Email,CHARINDEX('@',Email)+1,
(LEN(Email) - CHARINDEX('@',Email))) as EmailDomain,
Count(*) as Total
From tblEmployee
Group By SUBSTRING(Email,CHARINDEX('@',Email)+1,
(LEN(Email) - CHARINDEX('@',Email)))
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
3
0:39 / 14:56

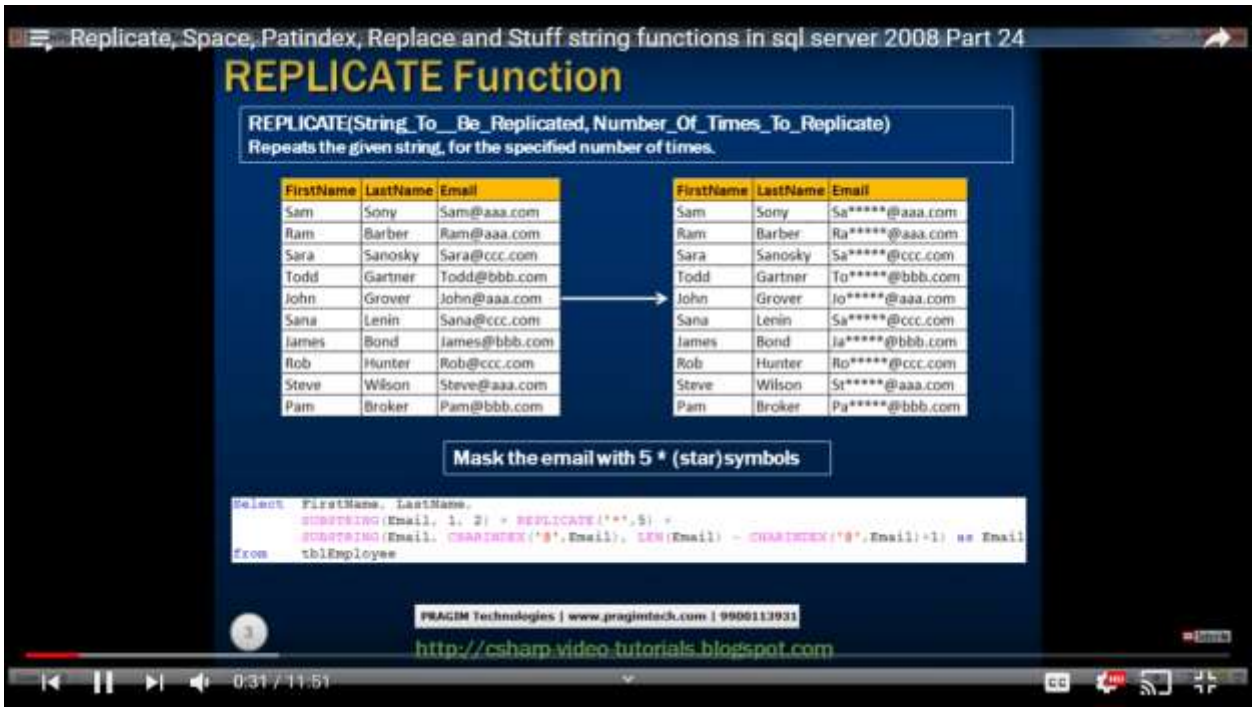

Select * from tblEmployee
SELECT RIGHT('ABCDEF', 3)
DEF

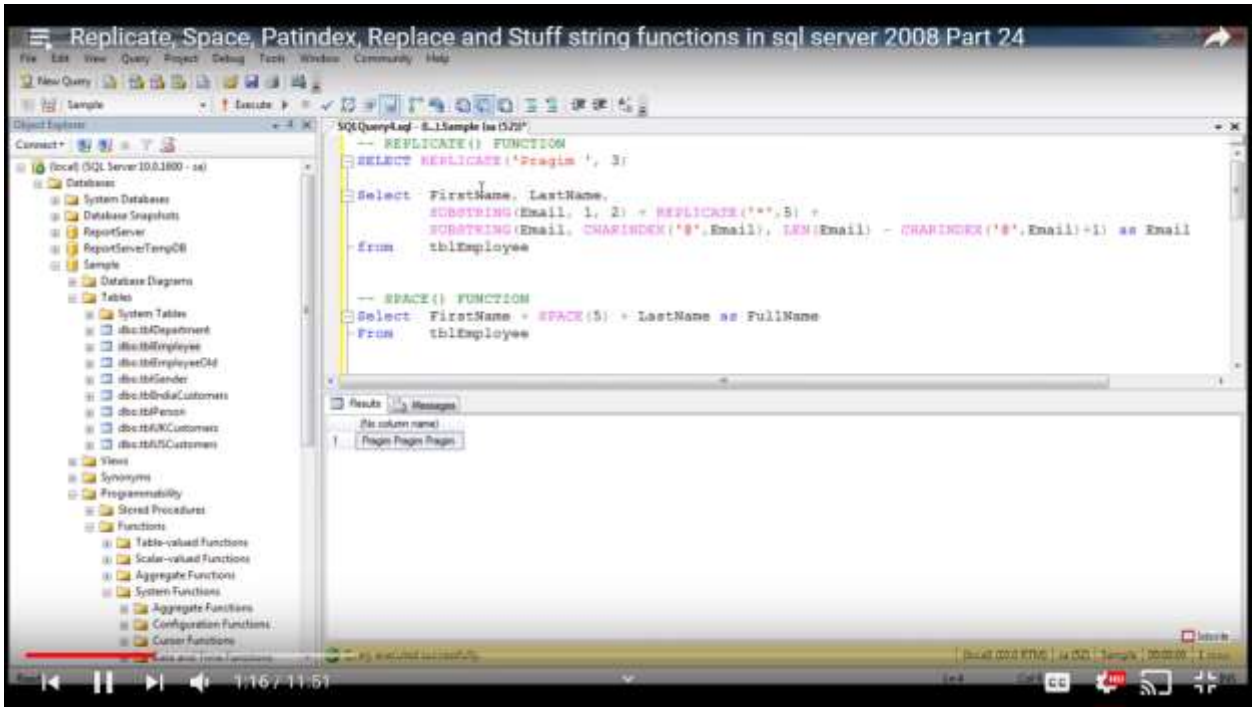

LEFT, RIGHT, CHARINDEX and SUBSTRING functions in sql server Part 23
Select * from tblEmployee
SELECT CHARINDEX('@', 'sara@aaa.com')
SELECT SUBSTRING('sara@aaa.com', 6, 3)
6:49 / 14:56

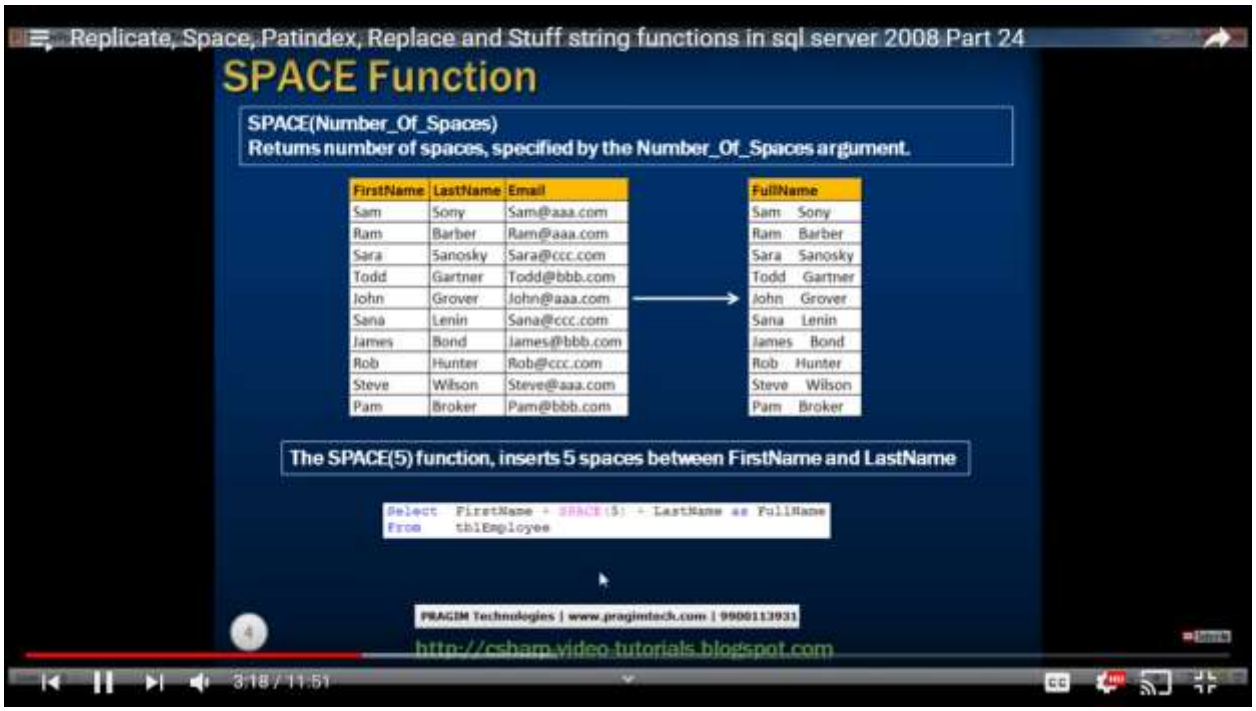

LEFT, RIGHT, CHARINDEX and SUBSTRING functions in sql server Part 23
Select * from tblEmployee
SELECT CHARINDEX('@', 'sara@aaa.com')
SELECT SUBSTRING('pam@bbb.com', CHARINDEX('@', 'pam@bbb.com') + 1,
LEN('pam@bbb.com') - CHARINDEX('@', 'pam@bbb.com'))
11:01 / 14:56


LEFT, RIGHT, CHARINDEX and SUBSTRING functions in sql server Part 23
Select * from tblEmployee
Select SUBSTRING(Email, CHARINDEX('@', Email) + 1,
LEN(Email) - CHARINDEX('@', Email)) as EmailDomain,
COUNT(Email) as Total
from tblEmployee
Group By SUBSTRING(Email, CHARINDEX('@', Email) + 1,
LEN(Email) - CHARINDEX('@', Email))
SELECT CHARINDEX('@', 'sara@aaa.com')
14:30 / 14:56

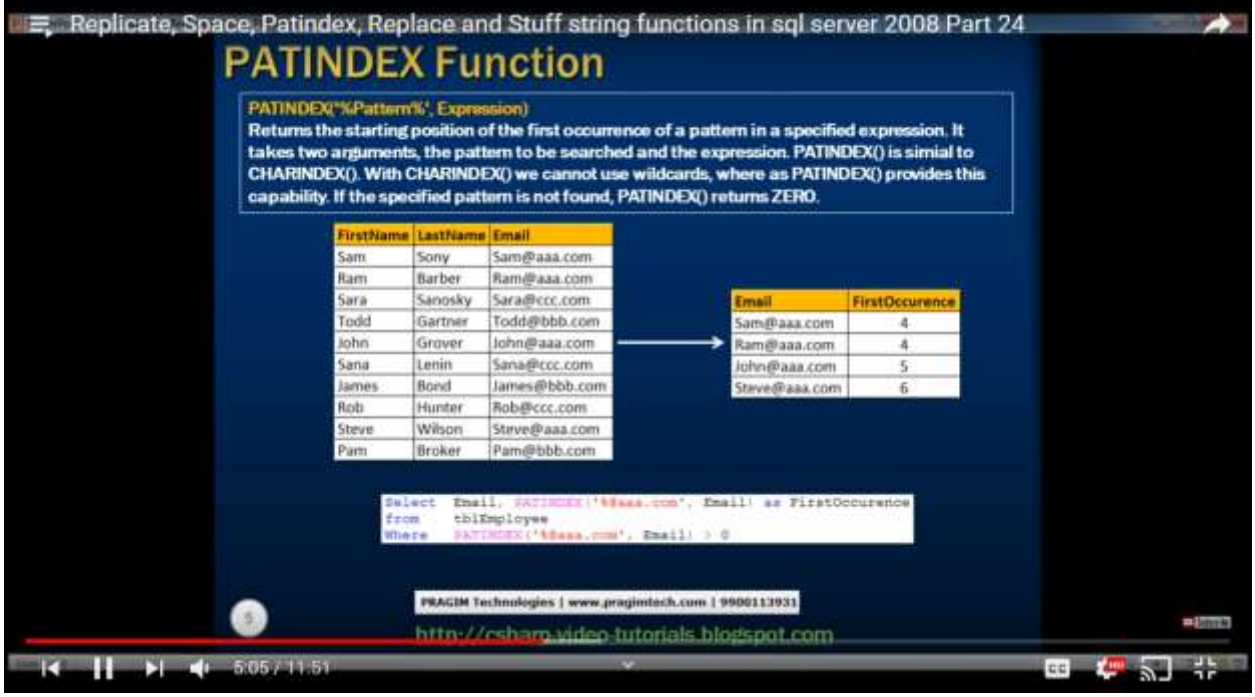

Replicate, Space, Patindex, Replace and Stuff string functions in sql server 2008 Part 24
In this session we will learn
Few more commonly used built-in string functions in SQL server
Replicate
Space
Patindex
Replace
Stuff
Pre-requisite:
Part 22 – Built in string functions in sql server
Part 23 – String functions continued
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
Exit full screen (f)
0:19 / 11:51

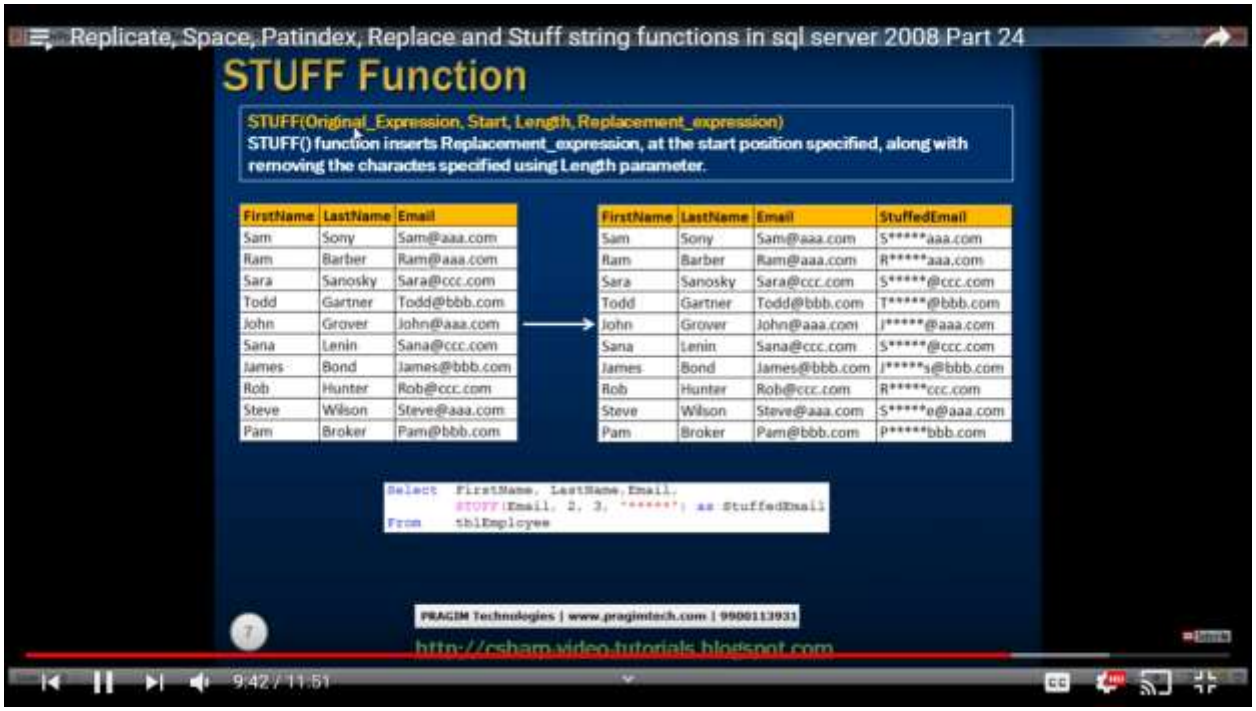

Replicate, Space, Patindex, Replace and Stuff string functions in sql server 2008 Part 24
REPLICATE Function
REPLICATE(String_To__Be_Replicated, Number_Of_Times_To_Replicate)
Repeats the given string, for the specified number of times.
FirstName LastName Email
Sam Sony Sam@aaa.com
Ram Barber Ram@aaa.com
Sara Sanosky Sara@ccc.com
Todd Gartner Todd@bbb.com
John Grover John@aaa.com
Sana Lenin Sana@ccc.com
James Bond James@bbb.com
Rob Hunter Rob@ccc.com
Steve Wilson Steve@aaa.com
Pam Broker Pam@bbb.com
FirstName LastName Email
Sam Sony Sa*****@aaa.com
Ram Barber Ra*****@aaa.com
Sara Sanosky Sa*****@ccc.com
Todd Gartner To*****@bbb.com
John Grover Jo*****@aaa.com
Sana Lenin Sa*****@ccc.com
James Bond Ja*****@bbb.com
Rob Hunter Ro*****@ccc.com
Steve Wilson St*****@aaa.com
Pam Broker Pa*****@bbb.com
Mask the email with 5 * (star) symbols
Select FirstName, LastName,
SUBSTRING(Email, 1, 2) + REPLICATE('*',5) +
SUBSTRING(Email, CHARINDEX('@',Email), LEN(Email) - CHARINDEX('@',Email)+1) as Email
from tblEmployee
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:31 / 11:51

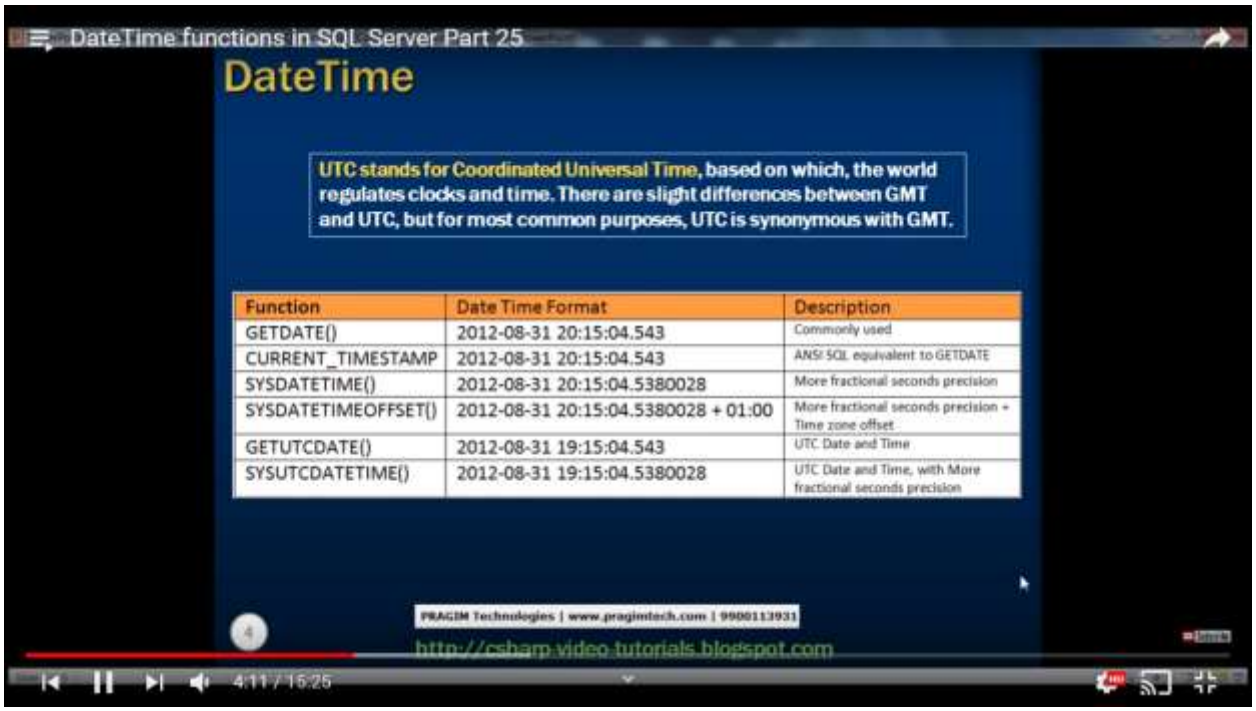

Replicate, Space, Patindex, Replace and Stuff string functions in sql server 2008 Part 24
-- REPLICATE() FUNCTION
SELECT REPLICATE('Pragim ', 3);
Select FirstName, LastName,
SUBSTRING(Email, 1, 2) + REPLICATE('*',5) +
SUBSTRING(Email, CHARINDEX('@',Email), LEN(Email) - CHARINDEX('@',Email)+1) as Email
from tblEmployee
-- SPACE() FUNCTION
Select FirstName + SPACE(5) + LastName as FullName
From tblEmployee
1:16 / 11:51

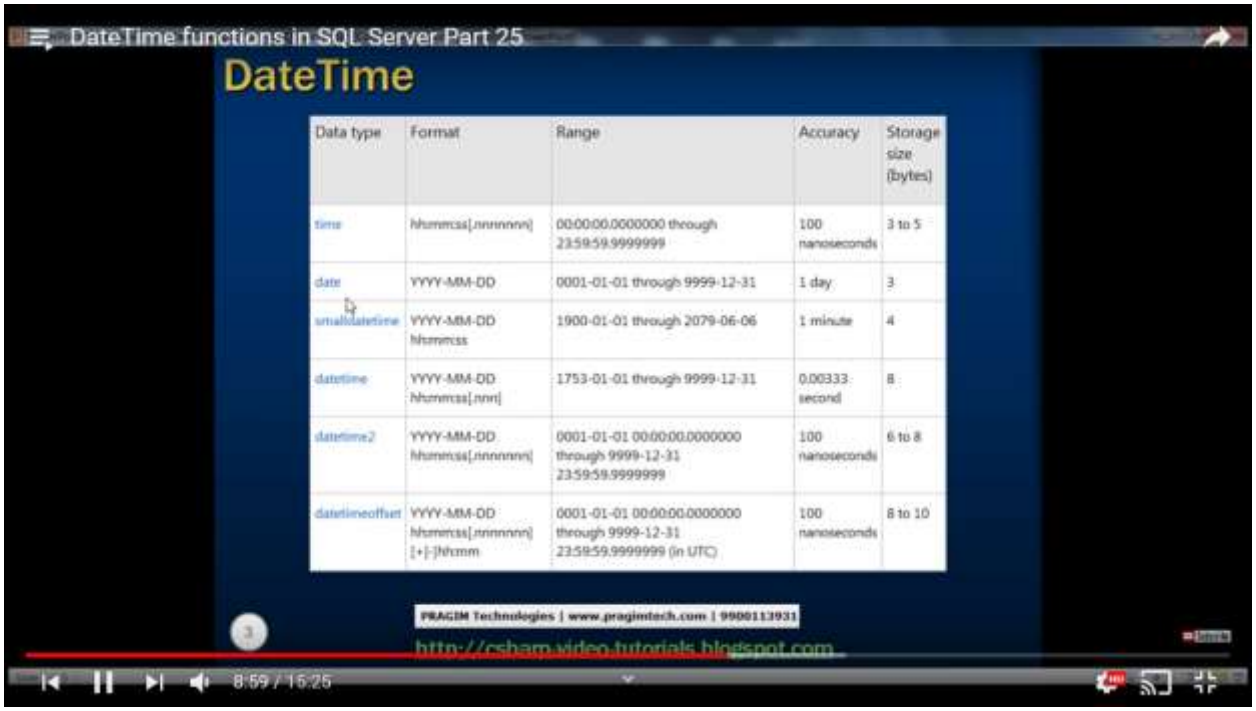

Replicate, Space, Patindex, Replace and Stuff string functions in sql server 2008 Part 24
SPACE Function
SPACE(Number_Of_Spaces)
Returns number of spaces, specified by the Number_Of_Spaces argument.
FirstName LastName Email
Sam Sony Sam@aaa.com
Ram Barber Ram@aaa.com
Sara Sanosky Sara@ccc.com
Todd Gartner Todd@bbb.com
John Grover John@aaa.com
Sana Lenin Sana@ccc.com
James Bond James@bbb.com
Rob Hunter Rob@ccc.com
Steve Wilson Steve@aaa.com
Pam Broker Pam@bbb.com
FullName
Sam Sony
Ram Barber
Sara Sanosky
Todd Gartner
John Grover
Sana Lenin
James Bond
Rob Hunter
Steve Wilson
Pam Broker
The SPACE(5) function, inserts 5 spaces between FirstName and LastName
Select FirstName + SPACE(5) + LastName as FullName
From tblEmployee
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
3:18 / 11:51


REPLACE Function
REPLACE(String_Expression, Pattern , Replacement_Value)
Replaces all occurrences of a specified string value with another string value.
FirstName | LastName | Email
Sam | Sony | Sam@aaa.com
Ram | Barber | Ram@aaa.com
Sara | Sanosky | Sara@ccc.com
Todd | Gartner | Todd@bbb.com
John | Grover | John@aaa.com
Sana | Lenin | Sana@ccc.com
James | Bond | James@bbb.com
Rob | Hunter | Rob@ccc.com
Steve | Wilson | Steve@aaa.com
Pam | Broker | Pam@bbb.com
Email | ConvertedEmail
Sam@aaa.com | Sam@aaa.net
Ram@aaa.com | Ram@aaa.net
Sara@ccc.com | Sara@ccc.net
Todd@bbb.com | Todd@bbb.net
John@aaa.com | John@aaa.net
Sana@ccc.com | Sana@ccc.net
James@bbb.com | James@bbb.net
Rob@ccc.com | Rob@ccc.net
Steve@aaa.com | Steve@aaa.net
Pam@bbb.com | Pam@bbb.net
All .COM strings are replaced with .NET
Select Email, REPLACE(Email, '.com', '.net') as ConvertedEmail
from tblEmployee
PRAGIM Technologies | www.pragimtech.com | 9900113931
6
http://csharp-video-tutorials.blogspot.com

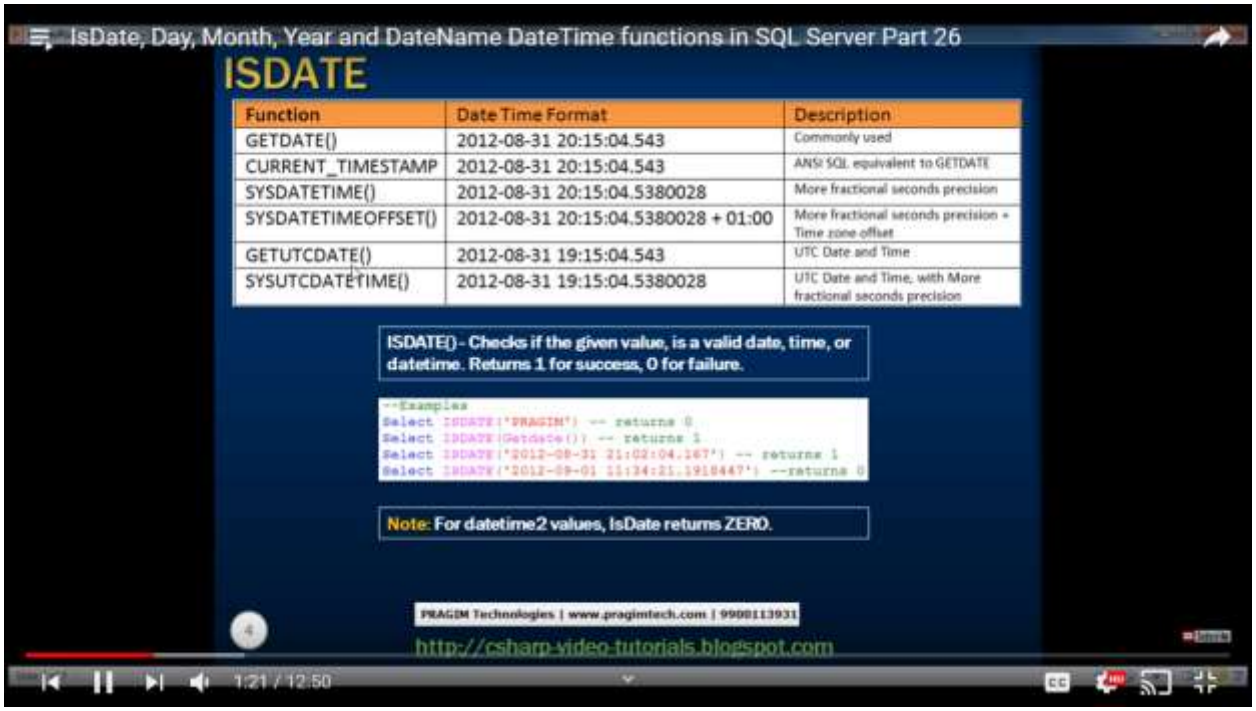

Replicate, Space, Patindex, Replace and Stuff string functions in sql server 2008 Part 24
PATINDEX Function
PATINDEX('%Pattern%', Expression)
Returns the starting position of the first occurrence of a pattern in a specified expression. It takes two arguments, the pattern to be searched and the expression. PATINDEX() is simial to CHARINDEX(). With CHARINDEX() we cannot use wildcards, where as PATINDEX() provides this capability. If the specified pattern is not found, PATINDEX() returns ZERO.
FirstName | LastName | Email
Sam | Sony | Sam@aaa.com
Ram | Barber | Ram@aaa.com
Sara | Sanosky | Sara@ccc.com
Todd | Gartner | Todd@bbb.com
John | Grover | John@aaa.com
Sana | Lenin | Sana@ccc.com
James | Bond | James@bbb.com
Rob | Hunter | Rob@ccc.com
Steve | Wilson | Steve@aaa.com
Pam | Broker | Pam@bbb.com
Email | FirstOccurence
Sam@aaa.com | 4
Ram@aaa.com | 4
John@aaa.com | 5
Steve@aaa.com | 6
Select Email, PATINDEX('%@aaa.com', Email) as FirstOccurence
from tblEmployee
Where PATINDEX('%@aaa.com', Email) > 0
PRAGIM Technologies | www.pragimtech.com | 9900113931
5
http://csharp-video-tutorials.blogspot.com
5:05 / 11:51

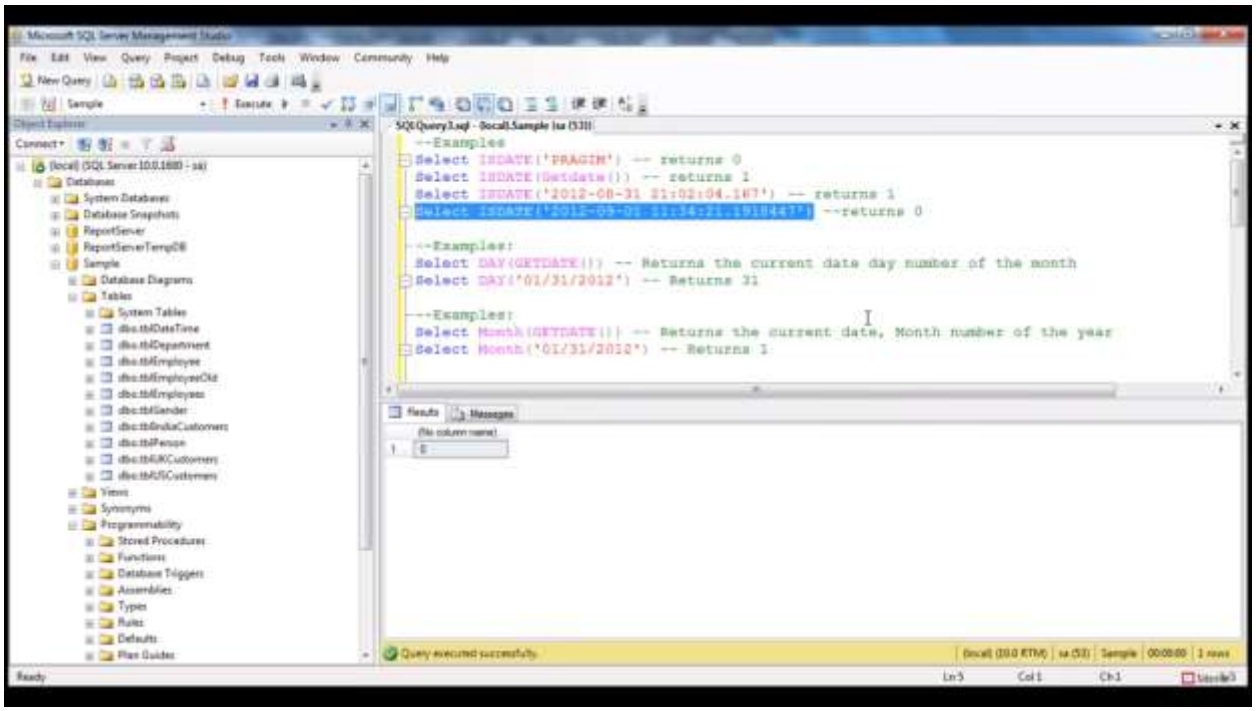

Replicate, Space, Patindex, Replace and Stuff string functions in sql server 2008 Part 24
STUFF Function
STUFF(Original_Expression, Start, Length, Replacement_expression)
STUFF() function inserts Replacement_expression, at the start position specified, along with removing the charactes specified using Length parameter.
FirstName LastName Email
Sam Sony Sam@aaa.com
Ram Barber Ram@aaa.com
Sara Sanosky Sara@ccc.com
Todd Gartner Todd@bbb.com
John Grover John@aaa.com
Sana Lenin Sana@ccc.com
James Bond James@bbb.com
Rob Hunter Rob@ccc.com
Steve Wilson Steve@aaa.com
Pam Broker Pam@bbb.com
FirstName LastName Email StuffedEmail
Sam Sony Sam@aaa.com S*****aaa.com
Ram Barber Ram@aaa.com R*****aaa.com
Sara Sanosky Sara@ccc.com S*****@ccc.com
Todd Gartner Todd@bbb.com T*****@bbb.com
John Grover John@aaa.com J*****@aaa.com
Sana Lenin Sana@ccc.com S*****@ccc.com
James Bond James@bbb.com J*****s@bbb.com
Rob Hunter Rob@ccc.com R*****ccc.com
Steve Wilson Steve@aaa.com S*****e@aaa.com
Pam Broker Pam@bbb.com P*****bbb.com
Select FirstName, LastName,Email,
STUFF(Email, 2, 3, '*****') as StuffedEmail
From tblEmployee
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
9:42 / 11:51

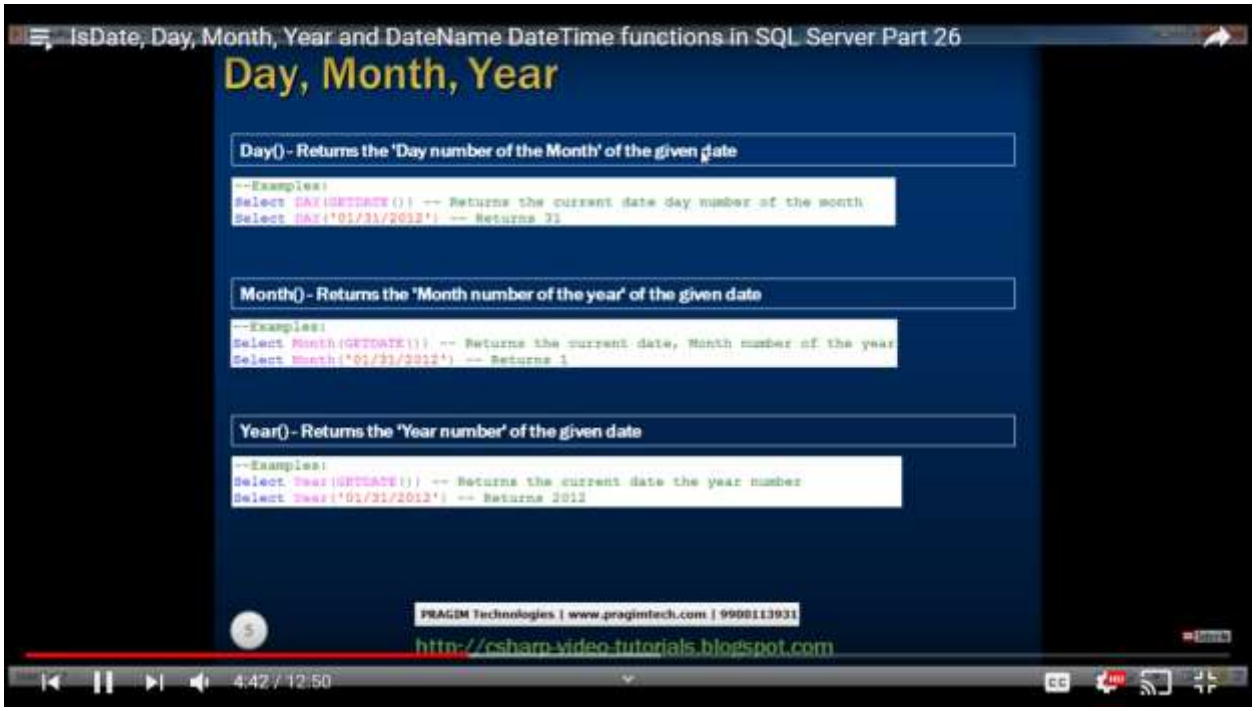

DateTime functions in SQL Server Part 25
DateTime
UTC stands for Coordinated Universal Time, based on which, the world regulates clocks and time. There are slight differences between GMT and UTC, but for most common purposes, UTC is synonymous with GMT.
Function Date Time Format Description
GETDATE() 2012-08-31 20:15:04.543 Commonly used
CURRENT_TIMESTAMP 2012-08-31 20:15:04.543 ANSI SQL equivalent to GETDATE
SYSDATETIME() 2012-08-31 20:15:04.5380028 More fractional seconds precision
SYSDATETIMEOFFSET() 2012-08-31 20:15:04.5380028 + 01:00 More fractional seconds precision + Time zone offset
GETUTCDATE() 2012-08-31 19:15:04.543 UTC Date and Time
SYSUTCDATETIME() 2012-08-31 19:15:04.5380028 UTC Date and Time, with More fractional seconds precision
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
4:11 / 16:25

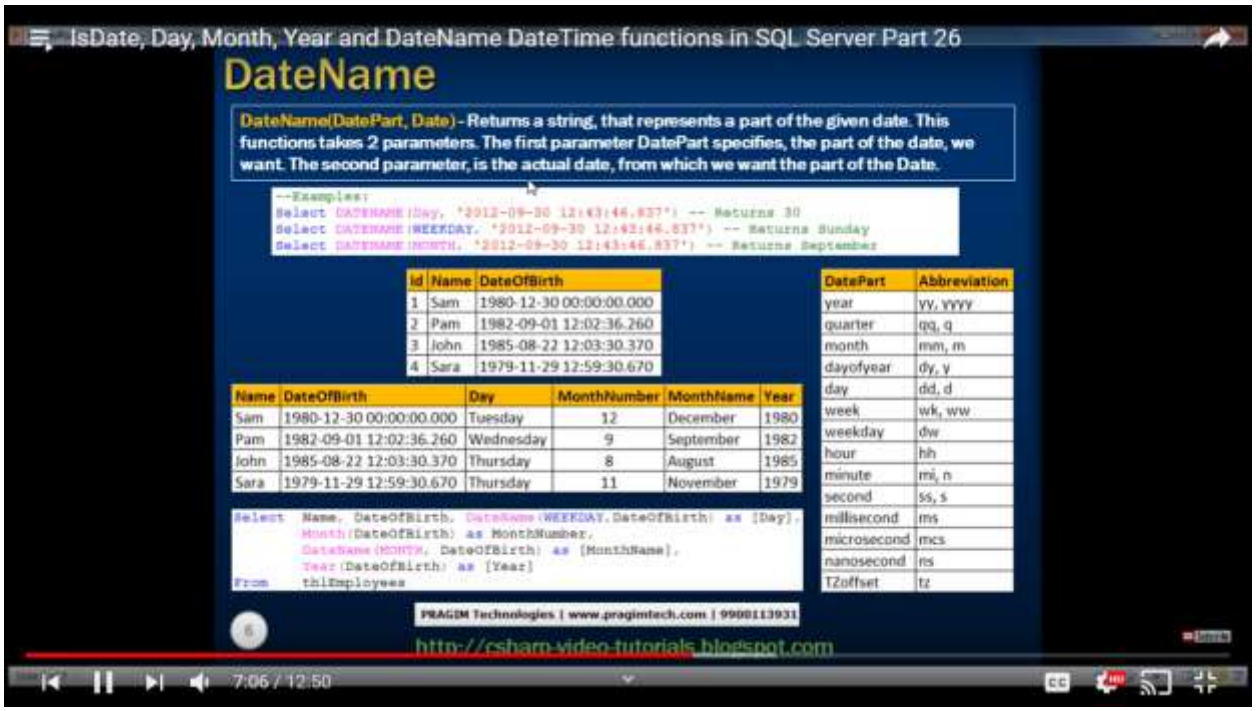

DateTime functions in SQL Server Part 25
DateTime
Data type | Format | Range | Accuracy | Storage size (bytes)
time | hh:mm:ss[.nnnnnnn] | 00:00:00.0000000 through 23:59:59.9999999 | 100 nanoseconds | 3 to 5
date | YYYY-MM-DD | 0001-01-01 through 9999-12-31 | 1 day | 3
smalldatetime | YYYY-MM-DD hh:mm:ss | 1900-01-01 through 2079-06-06 | 1 minute | 4
datetime | YYYY-MM-DD hh:mm:ss[.nnn] | 1753-01-01 through 9999-12-31 | 0.00333 second | 8
datetime2 | YYYY-MM-DD hh:mm:ss[.nnnnnnn] | 0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999 | 100 nanoseconds | 6 to 8
datetimeoffset | YYYY-MM-DD hh:mm:ss[.nnnnnnn] [+-]hh:mm | 0001-01-01 00:00:00.0000000 through 9999-12-31 23:59:59.9999999 (in UTC) | 100 nanoseconds | 8 to 10
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp.video.tutorials.blogspot.com
8:59 / 15:25

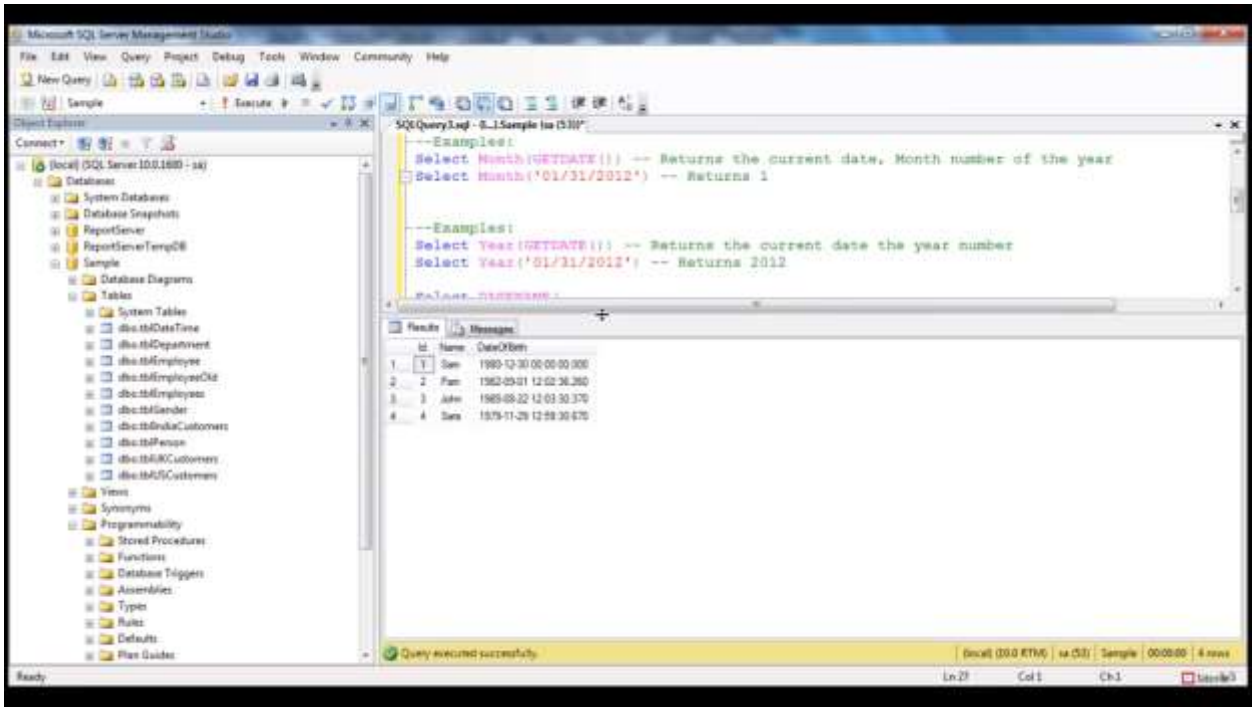

DateTime functions in SQL Server Part 25
SELECT * FROM tblDateTime
INSERT INTO tblDateTime VALUES (GETDATE(),GETDATE(),GETDATE(),GETDATE(),GETDATE(),GETDATE
UPDATE tblDateTime SET c_datetimeoffset = '2012-08-31 22:22:46.0500000 +02:00'
WHERE c_datetimeoffset = '2012-08-31 21:34:36.9200000 +00:00'
SELECT GETDATE(), 'GETDATE()'
SELECT CURRENT_TIMESTAMP, 'CURRENT_TIMESTAMP'
SELECT SYSDATETIME(), 'SYSDATETIME()'
SELECT SYSDATETIMEOFFSET(), 'SYSDATETIMEOFFSET()'
SELECT GETUTCDATE(), 'GETUTCDATE()'
11:51 / 15:25

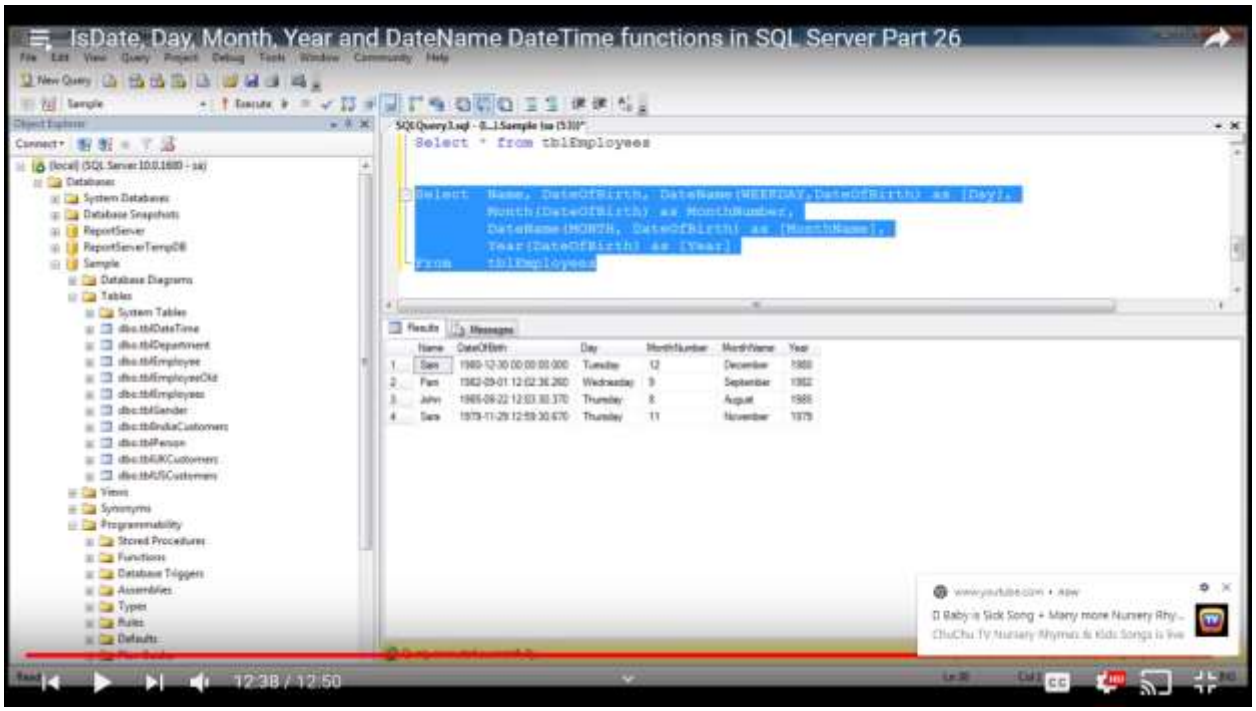

IsDate, Day, Month, Year and DateName DateTime functions in SQL Server Part 26
ISDATE
Function | Date Time Format | Description
GETDATE() | 2012-08-31 20:15:04.543 | Commonly used
CURRENT_TIMESTAMP | 2012-08-31 20:15:04.543 | ANSI SQL equivalent to GETDATE
SYSDATETIME() | 2012-08-31 20:15:04.5380028 | More fractional seconds precision
SYSDATETIMEOFFSET() | 2012-08-31 20:15:04.5380028 + 01:00 | More fractional seconds precision + Time zone offset
GETUTCDATE() | 2012-08-31 19:15:04.543 | UTC Date and Time
SYSUTCDATETIME() | 2012-08-31 19:15:04.5380028 | UTC Date and Time, with More fractional seconds precision
ISDATE() - Checks if the given value, is a valid date, time, or datetime. Returns 1 for success, 0 for failure.
--Examples
Select ISDATE('PRAGIM') -- returns 0
Select ISDATE(Getdate()) -- returns 1
Select ISDATE('2012-08-31 21:02:04.167') -- returns 1
Select ISDATE('2012-09-01 11:34:21.1918447') --returns 0
Note: For datetime2 values, IsDate returns ZERO.
PRAGIM Technologies | www.pragimtech.com | 9908113931
http://csharp-video-tutorials.blogspot.com
1:21 / 12:50

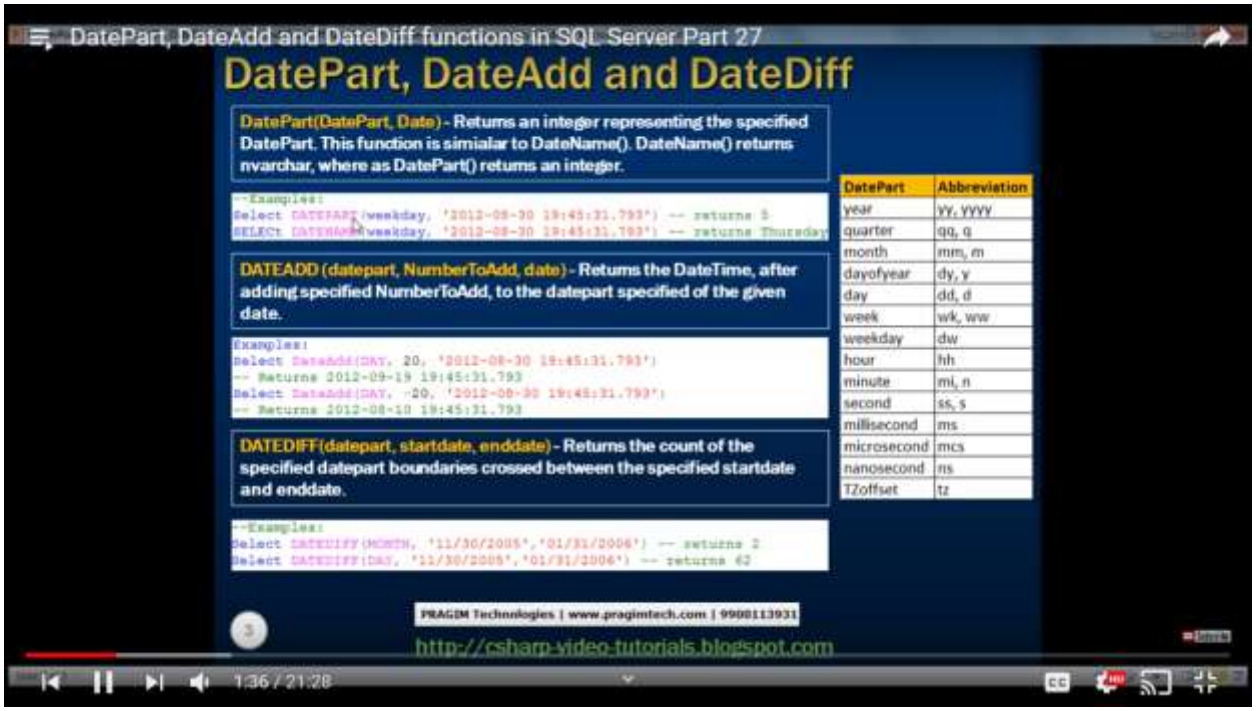

--Examples
Select ISDATE('PRAGIM') -- returns 0
Select ISDATE(Getdate()) -- returns 1
Select ISDATE('2012-08-31 21:02:04.167') -- returns 1
Select ISDATE('2012-09-01 11:34:21.1918447') --returns 0
--Examples:
Select DAY(GETDATE()) -- Returns the current date day number of the month
Select DAY('01/31/2012') -- Returns 31
--Examples:
Select Month(GETDATE()) -- Returns the current date, Month number of the year
Select Month('01/31/2012') -- Returns 1


IsDate, Day, Month, Year and DateName DateTime functions in SQL Server Part 26
Day, Month, Year
Day() - Returns the 'Day number of the Month' of the given date
--Examples:
Select DAY(GETDATE()) -- Returns the current date day number of the month
Select DAY('01/31/2012') -- Returns 31
Month() - Returns the 'Month number of the year' of the given date
--Examples:
Select Month(GETDATE()) -- Returns the current date, Month number of the year
Select Month('01/31/2012') -- Returns 1
Year() - Returns the 'Year number' of the given date
--Examples:
Select Year(GETDATE()) -- Returns the current date the year number
Select Year('01/31/2012') -- Returns 2012
PRAGIM Technologies | www.pragimtech.com | 9908113931
http://csharp-video-tutorials.blogspot.com
4:42 / 12:50

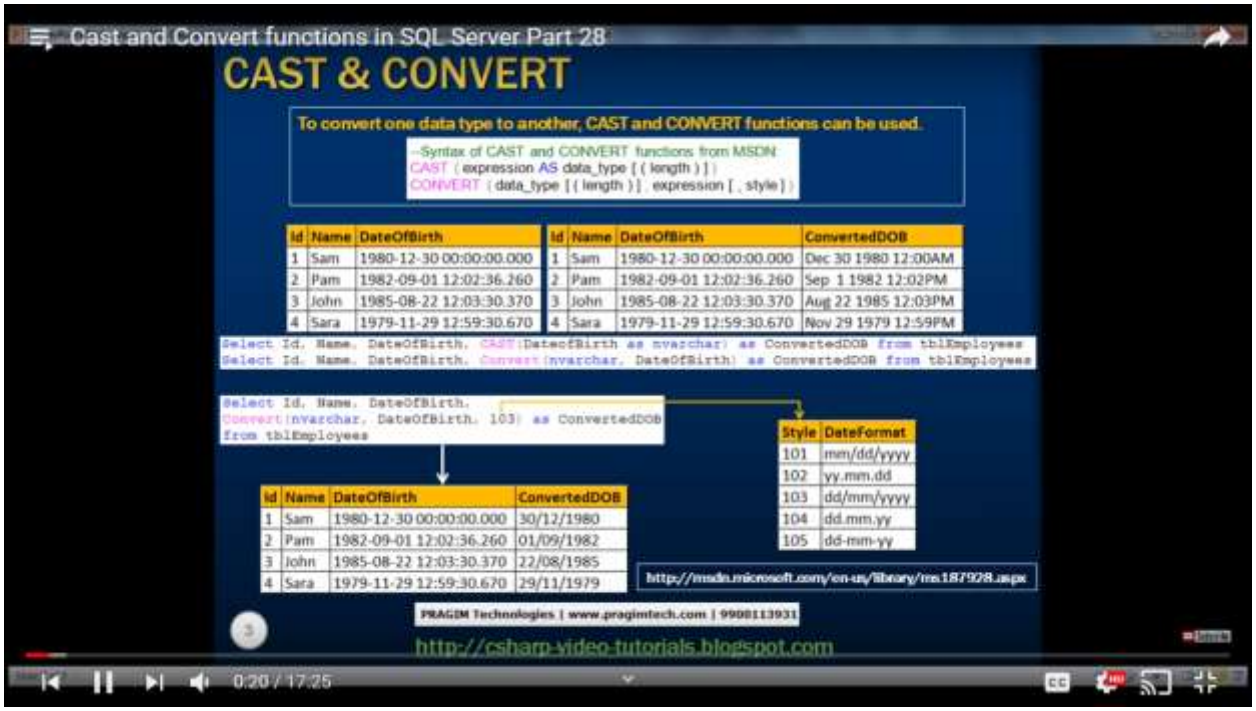

IsDate, Day, Month, Year and DateName DateTime functions in SQL Server Part 26
DateName
DateName(DatePart, Date) - Returns a string, that represents a part of the given date. This functions takes 2 parameters. The first parameter DatePart specifies, the part of the date, we want. The second parameter, is the actual date, from which we want the part of the Date.
--Examples:
Select DATENAME(Day, '2012-09-30 12:43:46.837') -- Returns 30
Select DATENAME(WEEKDAY, '2012-09-30 12:43:46.837') -- Returns Sunday
Select DATENAME(MONTH, '2012-09-30 12:43:46.837') -- Returns September
Id Name DateOfBirth
1 Sam 1980-12-30 00:00:00.000
2 Pam 1982-09-01 12:02:36.260
3 John 1985-08-22 12:03:30.370
4 Sara 1979-11-29 12:59:30.670
Name DateOfBirth Day MonthNumber MonthName Year
Sam 1980-12-30 00:00:00.000 Tuesday 12 December 1980
Pam 1982-09-01 12:02:36.260 Wednesday 9 September 1982
John 1985-08-22 12:03:30.370 Thursday 8 August 1985
Sara 1979-11-29 12:59:30.670 Thursday 11 November 1979
Select Name, DateOfBirth, DateName(WEEKDAY,DateOfBirth) as [Day],
Month(DateOfBirth) as MonthNumber,
DateName(MONTH, DateOfBirth) as [MonthName],
Year(DateOfBirth) as [Year]
From thlEmployees
DatePart Abbreviation
year yy, yyyy
quarter qq, q
month mm, m
dayofyear dy, y
day dd, d
week wk, ww
weekday dw
hour hh
minute mi, n
second ss, s
millisecond ms
microsecond mcs
nanosecond ns
TZoffset tz
PRAGIM Technologies | www.pragimtech.com | 9908113931
http://csharp-video-tutorials.blogspot.com
7:06 / 12:50

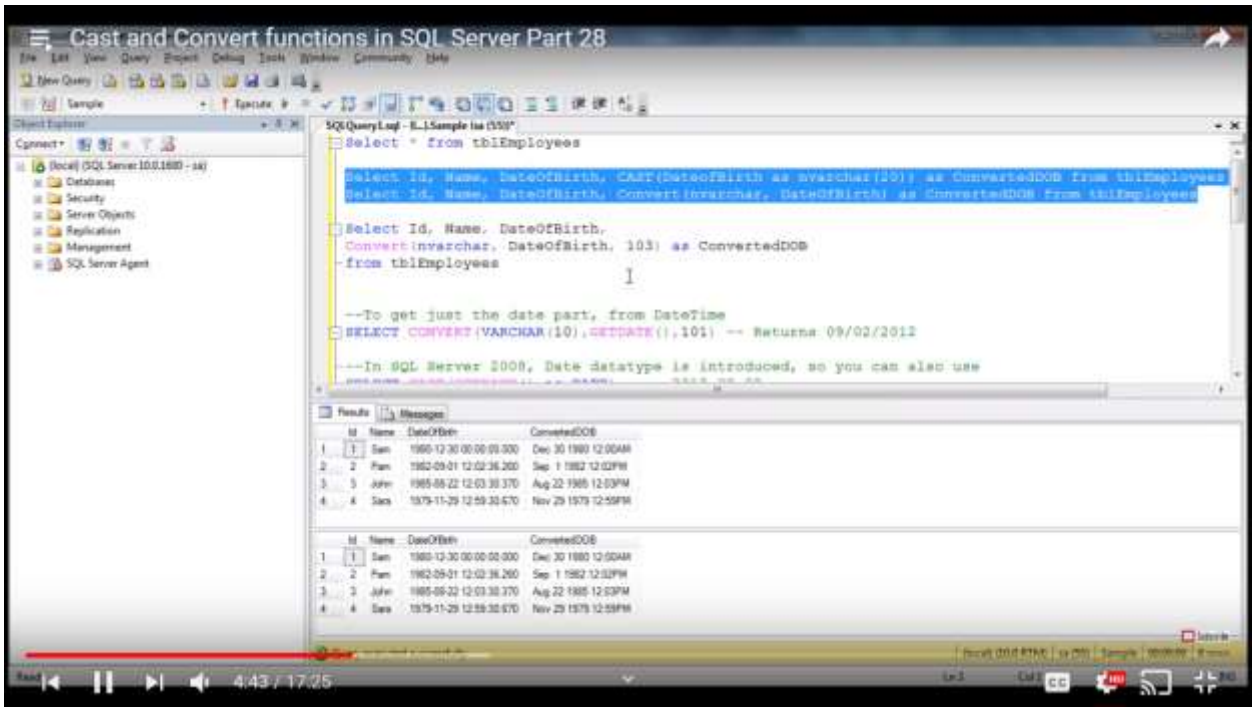

--Examples:
Select Month(GETDATE()) -- Returns the current date, Month number of the year
Select Month('01/31/2012') -- Returns 1
--Examples:
Select Year(GETDATE()) -- Returns the current date the year number
Select Year('01/31/2012') -- Returns 2012

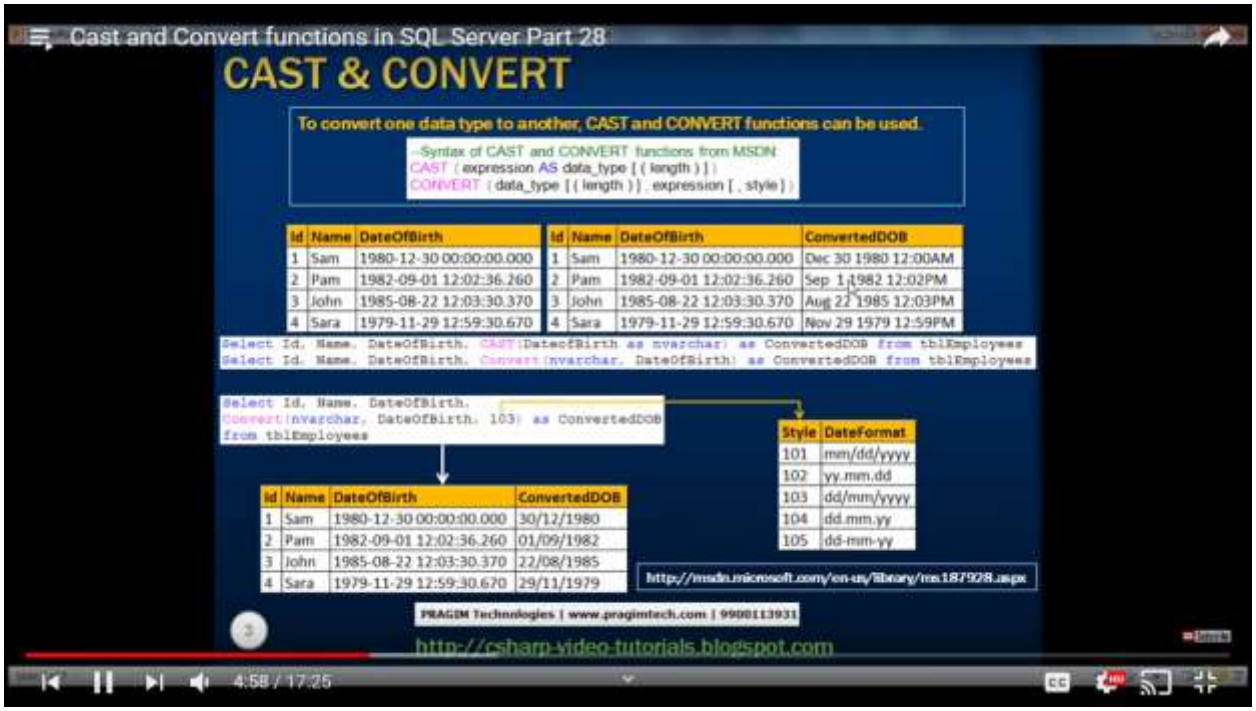

IsDate, Day, Month, Year and DateName DateTime functions in SQL Server Part 26
Select * from tblEmployees
Select Name, DateOfBirth, DateName(WEEKDAY,DateOfBirth) as [Day],
Month(DateOfBirth) as MonthNumber,
DateName(MONTH, DateOfBirth) as [MonthName],
Year(DateOfBirth) as [Year]
from tblEmployees
12:38 / 12:50

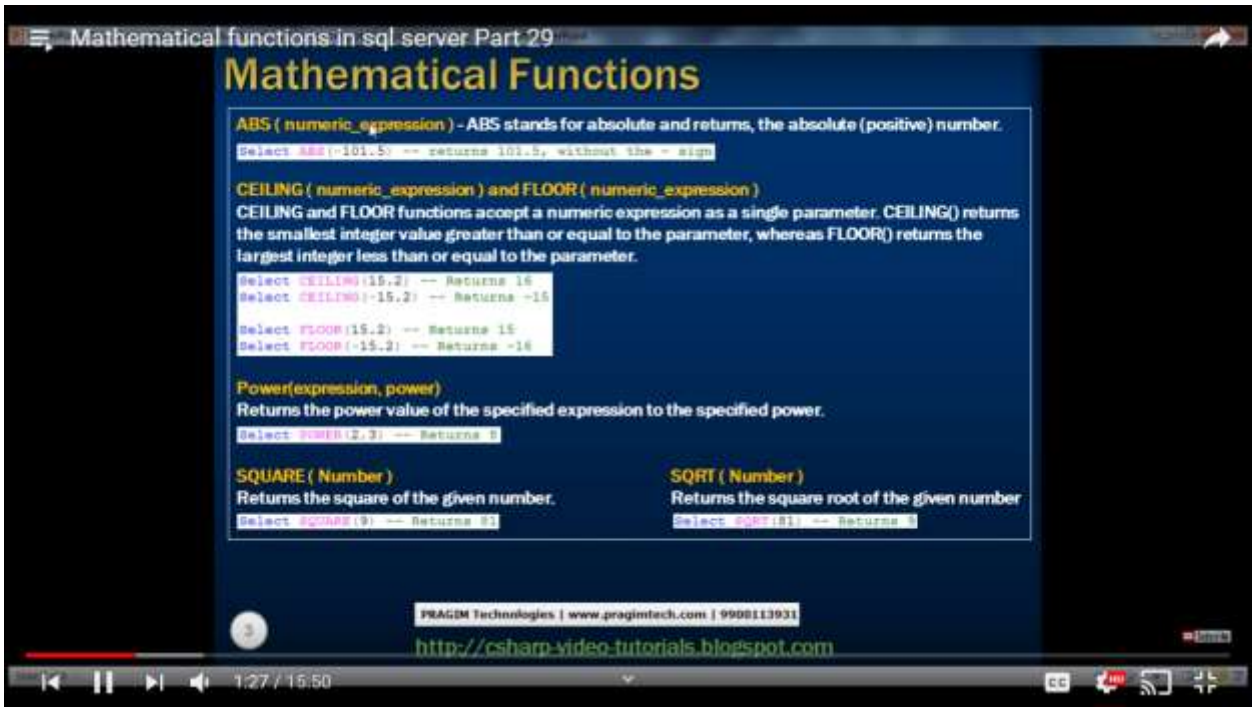

DatePart, DateAdd and DateDiff functions in SQL Server Part 27
DatePart, DateAdd and DateDiff
DatePart(DatePart, Date) - Returns an integer representing the specified DatePart. This function is simialar to DateName(). DateName() returns nvarchar, where as DatePart() returns an integer.
--Examples:
Select DATEPART(weekday, '2012-08-30 19:45:31.793') -- returns 5
SELECT DATENAME(weekday, '2012-08-30 19:45:31.793') -- returns Thursday
DATEADD (datepart, NumberToAdd, date) - Returns the DateTime, after adding specified NumberToAdd, to the datepart specified of the given date.
Examples:
Select Dateadd(DAY, 20, '2012-08-30 19:45:31.793')
-- Returns 2012-09-19 19:45:31.793
Select Dateadd(DAY, -20, '2012-08-30 19:45:31.793')
-- Returns 2012-08-10 19:45:31.793
DATEDIFF(datepart, startdate, enddate) - Returns the count of the specified datepart boundaries crossed between the specified startdate and enddate.
--Examples:
Select DATEDIFF(MONTH, '11/30/2005','01/31/2006') -- returns 2
Select DATEDIFF(DAY, '11/30/2005','01/31/2006') -- returns 62
DatePart Abbreviation
year yy, yyyy
quarter qq, q
month mm, m
dayofyear dy, y
day dd, d
week wk, ww
weekday dw
hour hh
minute mi, n
second ss, s
millisecond ms
microsecond mcs
nanosecond ns
TZoffset tz
PRAGIM Technologies | www.pragimtech.com | 9908113931
3
http://csharp-video-tutorials.blogspot.com
1:36 / 21:28

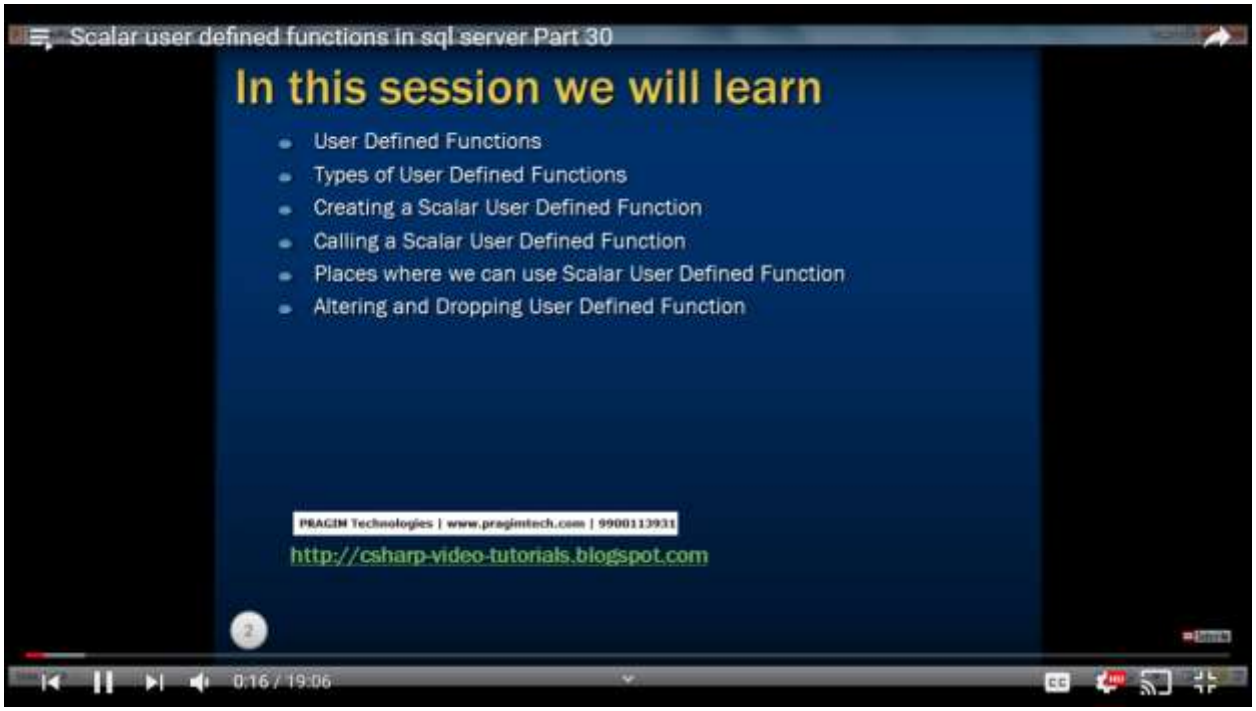

DatePart, DateAdd and DateDiff functions in SQL Server Part 27
Calculating Age
Id Name DateOfBirth
1 Sam 1980-12-30 00:00:00.000
2 Pam 1982-09-01 12:02:36.260
3 John 1985-08-22 12:03:30.370
4 Sara 1979-11-29 12:59:30.670
Id Name DateOfBirth DOB
1 Sam 1980-12-30 00:00:00.000 31 Years 8 Months 2 Days old
2 Pam 1982-09-01 12:02:36.260 30 Years 0 Months 0 Days old
3 John 1985-08-22 12:03:30.370 27 Years 0 Months 10 Days old
4 Sara 1979-11-29 12:59:30.670 32 Years 9 Months 3 Days old
DECLARE @DOB datetime, @tmpdate datetime, @years int, @months int, @days int
SET @DOB = '10/08/1982'
SELECT @tmpdate = @DOB
SELECT @years = DATEDIFF(YEAR, @tmpdate, GETDATE()) -
CASE
WHEN (MONTH(@DOB) > MONTH(GETDATE())) OR
(MONTH(@DOB) = MONTH(GETDATE()) AND DAY(@DOB) > DAY(GETDATE()))
THEN 1 ELSE 0
END
SELECT @tmpdate = DATEADD(YEAR, @years, @tmpdate)
SELECT @months = DATEDIFF(MONTH, @tmpdate, GETDATE()) -
CASE
WHEN DAY(@DOB) > DAY(GETDATE())
THEN 1 ELSE 0
END
SELECT @tmpdate = DATEADD(MONTH, @months, @tmpdate)
SELECT @days = DATEDIFF(DAY, @tmpdate, GETDATE())
SELECT @years as Years, @months as Months, @days as [Days]
PRAGIM Technologies | 9908113931
Scroll for details
4
http://csharp-video-tutorials.blogspot.com
7:19 / 21:28


Cast and Convert functions in SQL Server Part 28
CAST & CONVERT
To convert one data type to another, CAST and CONVERT functions can be used.
--Syntax of CAST and CONVERT functions from MSDN
CAST (expression AS data_type [(length)])
CONVERT (data_type [(length)] , expression [, style])
Id Name DateOfBirth
1 Sam 1980-12-30 00:00:00.000
2 Pam 1982-09-01 12:02:36.260
3 John 1985-08-22 12:03:30.370
4 Sara 1979-11-29 12:59:30.670
Id Name DateOfBirth ConvertedDOB
1 Sam 1980-12-30 00:00:00.000 Dec 30 1980 12:00AM
2 Pam 1982-09-01 12:02:36.260 Sep 1 1982 12:02PM
3 John 1985-08-22 12:03:30.370 Aug 22 1985 12:03PM
4 Sara 1979-11-29 12:59:30.670 Nov 29 1979 12:59PM
Select Id, Name, DateOfBirth, CAST(DateofBirth as nvarchar) as ConvertedDOB from tblEmployees
Select Id, Name, DateOfBirth, Convert(nvarchar, DateOfBirth) as ConvertedDOB from tblEmployees
Select Id, Name, DateOfBirth,
Convert(nvarchar, DateOfBirth, 103) as ConvertedDOB
from tblEmployees
Id Name DateOfBirth ConvertedDOB
1 Sam 1980-12-30 00:00:00.000 30/12/1980
2 Pam 1982-09-01 12:02:36.260 01/09/1982
3 John 1985-08-22 12:03:30.370 22/08/1985
4 Sara 1979-11-29 12:59:30.670 29/11/1979
Style DateFormat
101 mm/dd/yyyy
102 yy.mm.dd
103 dd/mm/yyyy
104 dd.mm.yy
105 dd-mm-yy
http://msdn.microsoft.com/en-us/library/ms187928.aspx
PRAGIM Technologies | www.pragimtech.com | 9908113931
http://csharp-video-tutorials.blogspot.com
0:20 / 17:25

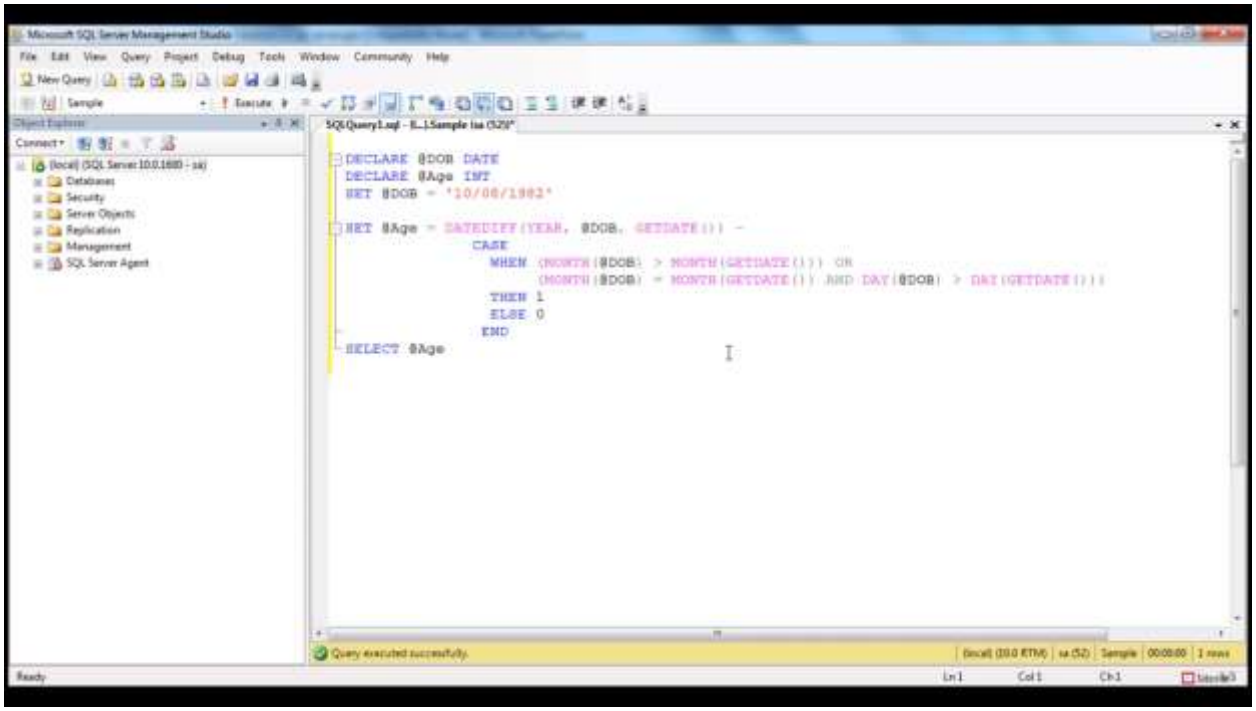

Cast and Convert functions in SQL Server Part 28
Select * from tblEmployees
Select Id, Name, DateOfBirth, CAST(DateofBirth as nvarchar(20)) as ConvertedDOB from tblEmployees
Select Id, Name, DateOfBirth, Convert(nvarchar, DateOfBirth) as ConvertedDOB from tblEmployees
Select Id, Name, DateOfBirth,
Convert(nvarchar, DateOfBirth, 103) as ConvertedDOB
from tblEmployees
--To get just the date part, from DateTime
SELECT CONVERT(VARCHAR(10),GETDATE(),101) -- Returns 09/02/2012
---In SQL Server 2008, Date datatype is introduced, so you can also use
4:43 / 17:25


Cast and Convert functions in SQL Server Part 28
CAST & CONVERT
To convert one data type to another, CAST and CONVERT functions can be used.
--Syntax of CAST and CONVERT functions from MSDN
CAST (expression AS data_type [(length)])
CONVERT (data_type [(length)] , expression [, style])
Id Name DateOfBirth
1 Sam 1980-12-30 00:00:00.000
2 Pam 1982-09-01 12:02:36.260
3 John 1985-08-22 12:03:30.370
4 Sara 1979-11-29 12:59:30.670
Id Name DateOfBirth ConvertedDOB
1 Sam 1980-12-30 00:00:00.000 Dec 30 1980 12:00AM
2 Pam 1982-09-01 12:02:36.260 Sep 1 1982 12:02PM
3 John 1985-08-22 12:03:30.370 Aug 22 1985 12:03PM
4 Sara 1979-11-29 12:59:30.670 Nov 29 1979 12:59PM
Select Id, Name, DateOfBirth, CAST(DateofBirth as nvarchar) as ConvertedDOB from tblEmployees
Select Id, Name, DateOfBirth, Convert(nvarchar, DateOfBirth) as ConvertedDOB from tblEmployees
Select Id, Name, DateOfBirth,
Convert(nvarchar, DateOfBirth, 103) as ConvertedDOB
from tblEmployees
Style DateFormat
101 mm/dd/yyyy
102 yy.mm.dd
103 dd/mm/yyyy
104 dd.mm.yy
105 dd-mm-yy
Id Name DateOfBirth ConvertedDOB
1 Sam 1980-12-30 00:00:00.000 30/12/1980
2 Pam 1982-09-01 12:02:36.260 01/09/1982
3 John 1985-08-22 12:03:30.370 22/08/1985
4 Sara 1979-11-29 12:59:30.670 29/11/1979
http://msdn.microsoft.com/en-us/library/ms187928.aspx
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
4:58 / 17:25

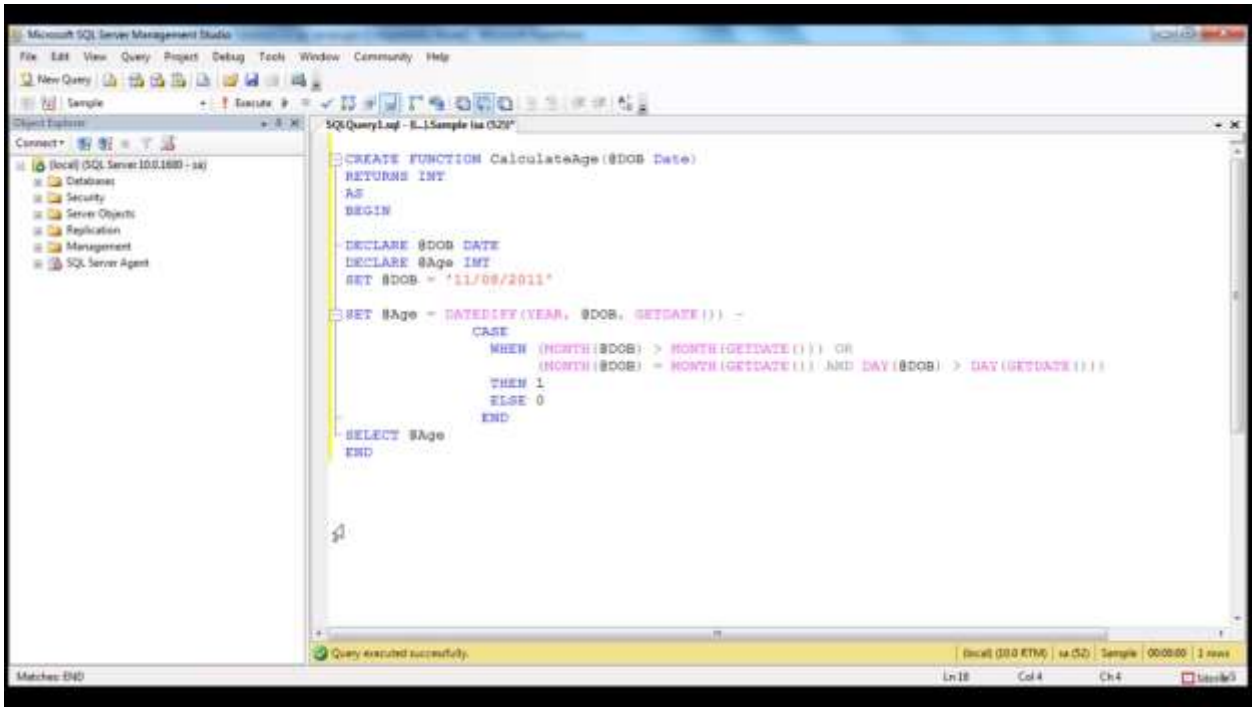

Mathematical functions in sql server Part 29
Mathematical Functions
ABS (numeric_expression) - ABS stands for absolute and returns, the absolute (positive) number.
Select ABS(-101.5) -- returns 101.5, without the - sign
CEILING (numeric_expression) and FLOOR (numeric_expression)
CEILING and FLOOR functions accept a numeric expression as a single parameter. CEILING() returns the smallest integer value greater than or equal to the parameter, whereas FLOOR() returns the largest integer less than or equal to the parameter.
Select CEILING(15.2) -- Returns 16
Select CEILING(-15.2) -- Returns -15
Select FLOOR(15.2) -- Returns 15
Select FLOOR(-15.2) -- Returns -16
Power(expression, power)
Returns the power value of the specified expression to the specified power.
Select POWER(2,3) -- Returns 8
SQUARE (Number)
Returns the square of the given number.
Select SQUARE(9) -- Returns 81
SQRT (Number)
Returns the square root of the given number
Select SQRT(81) -- Returns 9
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
1:27 / 15:50

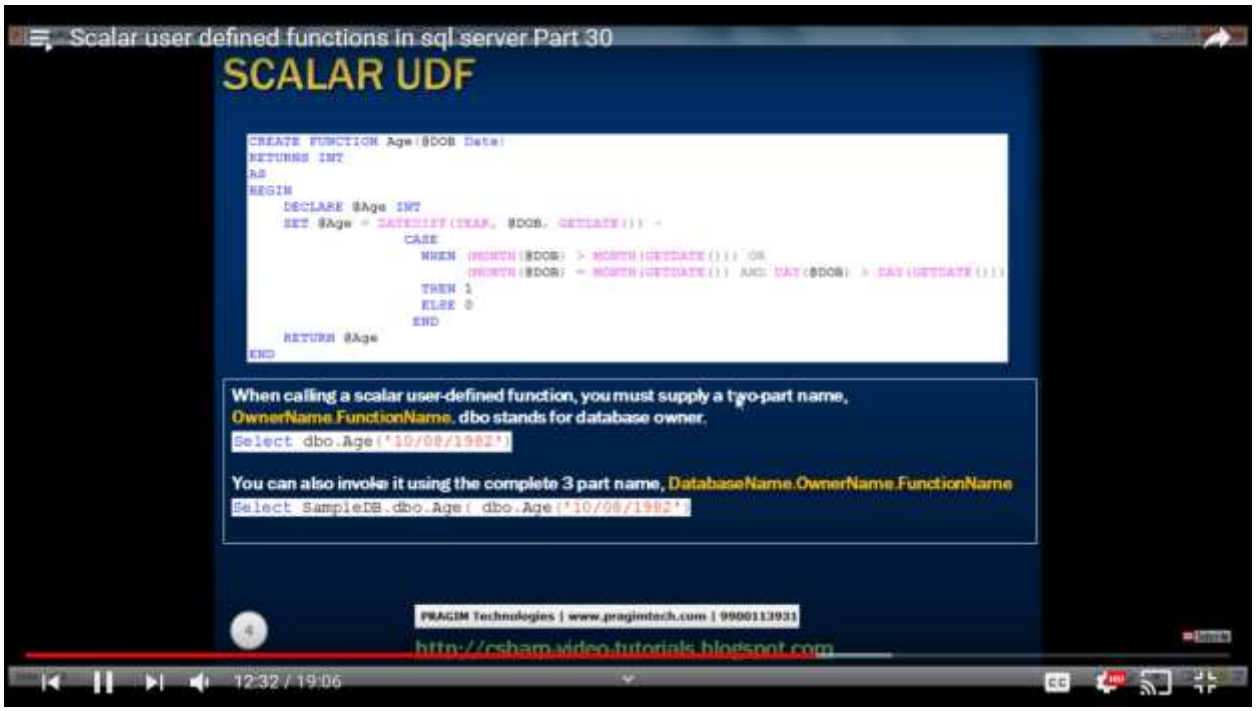

Scalar user defined functions in sql server Part 30
In this session we will learn
User Defined Functions
Types of User Defined Functions
Creating a Scalar User Defined Function
Calling a Scalar User Defined Function
Places where we can use Scalar User Defined Function
Altering and Dropping User Defined Function
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
2
0:16 / 19:06

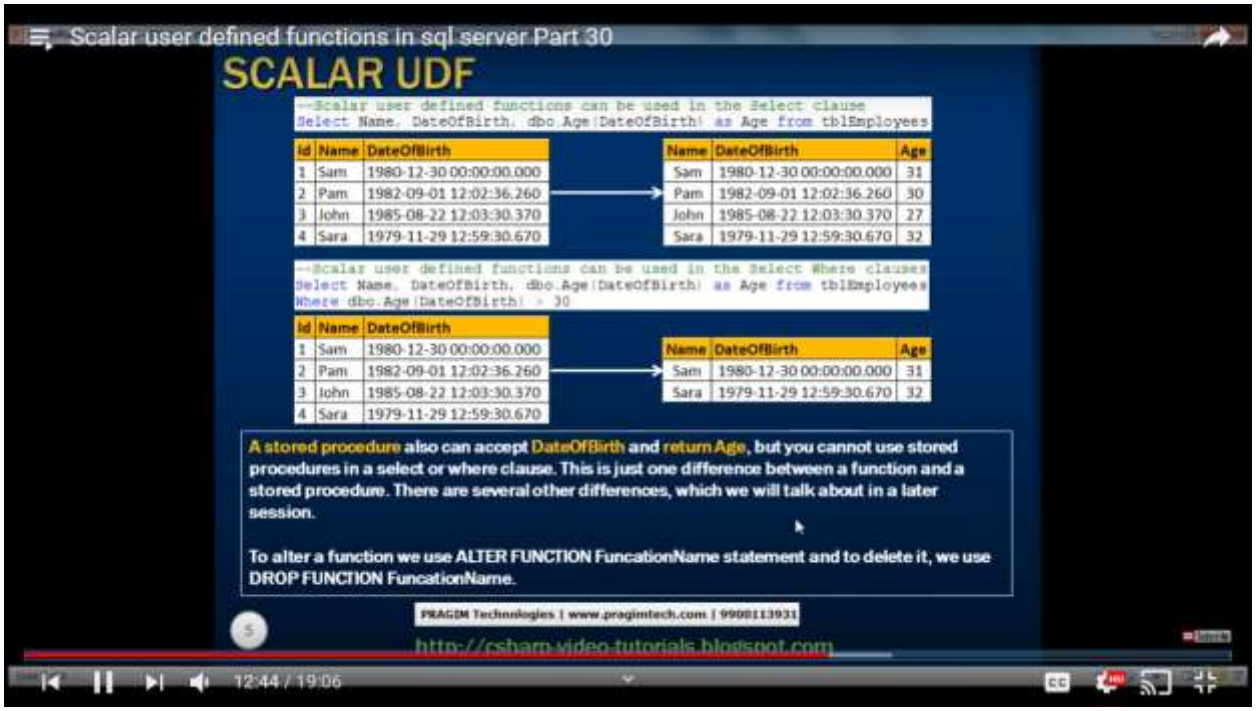

SCALAR UDF
From Parts 22 to 29, we have learnt how to use many of the system functions that are available in SQL Server. In this session, we will turn our attention, to creating user defined functions. In short UDF.
In SQL Server there are 3 types of User Defined functions
1. Scalar functions
2. Inline table-valued functions
3. Multi-statement table-valued functions
Scalar functions may or may not have parameters, but always return a single (scalar) value. The returned value can be of any data type, except text, ntext, image, cursor, and timestamp
--To create a function, we use the following syntax:
CREATE FUNCTION Function_Name(@Parameter1 DataType, @Parameter2 DataType, ... @ParameterN Datatype)
RETURNS Return_Datatype
AS
BEGIN
--Function Body
Return Return_Datatype
END
PRAGIM Technologies | www.pragimtech.com | 9900113931
3
http://csharp-video-tutorials.blogspot.com

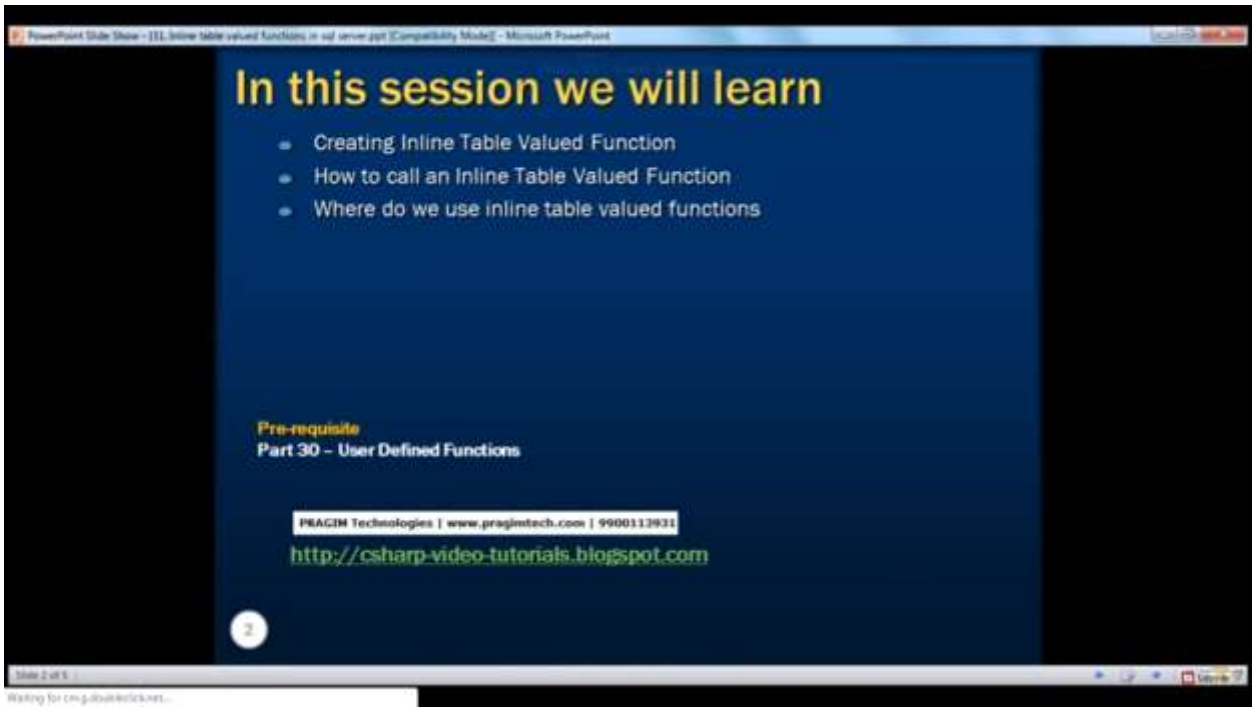

DECLARE @DOB DATE
DECLARE @Age INT
SET @DOB = '10/08/1982'
SET @Age = DATEDIFF(YEAR, @DOB, GETDATE()) -
CASE
WHEN (MONTH(@DOB) > MONTH(GETDATE())) OR
(MONTH(@DOB) = MONTH(GETDATE()) AND DAY(@DOB) > DAY(GETDATE()))
THEN 1
ELSE 0
END
SELECT @Age

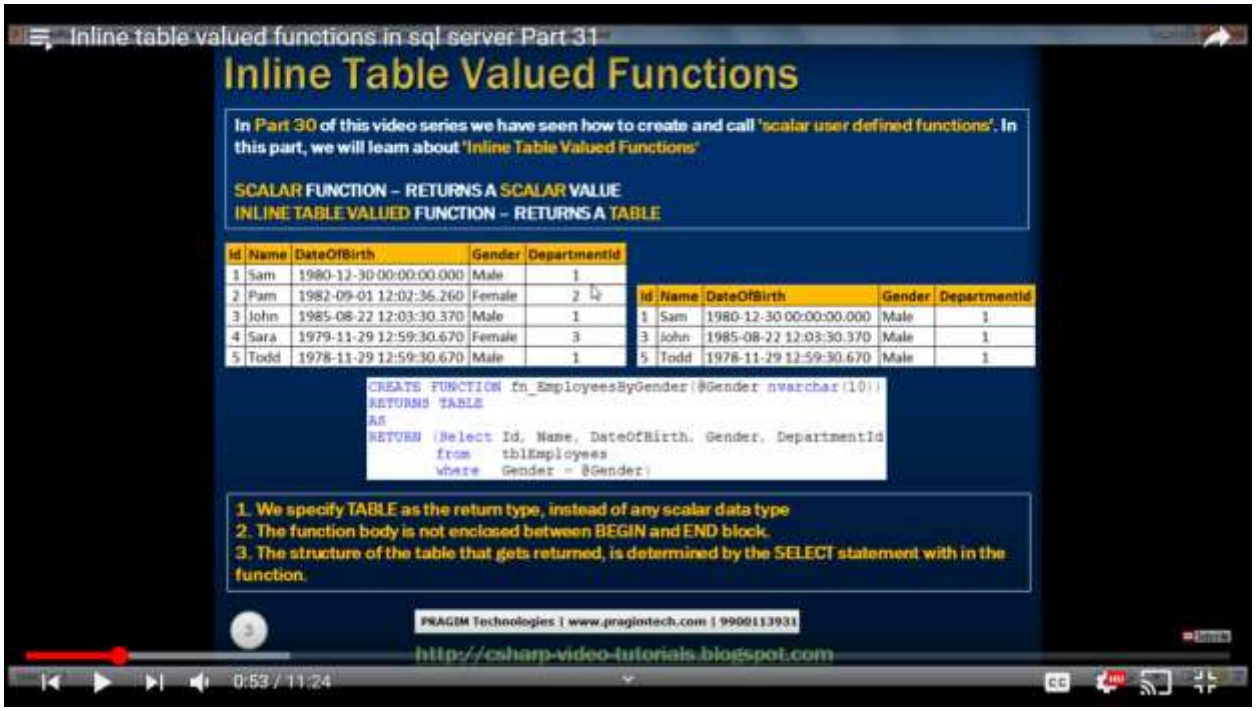

SCALAR UDF
CREATE FUNCTION Age(@DOB Date)
RETURNS INT
AS
BEGIN
DECLARE @Age INT
SET @Age = DATEDIFF(YEAR, @DOB, GETDATE()) -
CASE
WHEN (MONTH(@DOB) > MONTH(GETDATE())) OR
(MONTH(@DOB) = MONTH(GETDATE()) AND DAY(@DOB) > DAY(GETDATE()))
THEN 1
ELSE 0
END
RETURN @Age
END
When calling a scalar user-defined function, you must supply a two-part name, OwnerName.FunctionName. dbo stands for database owner.
Select dbo.Age('10/08/1982')
You can also invoke it using the complete 3 part name, DatabaseName.OwnerName.FunctionName
Select SampleDB.dbo.Age(dbo.Age('10/08/1982')
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

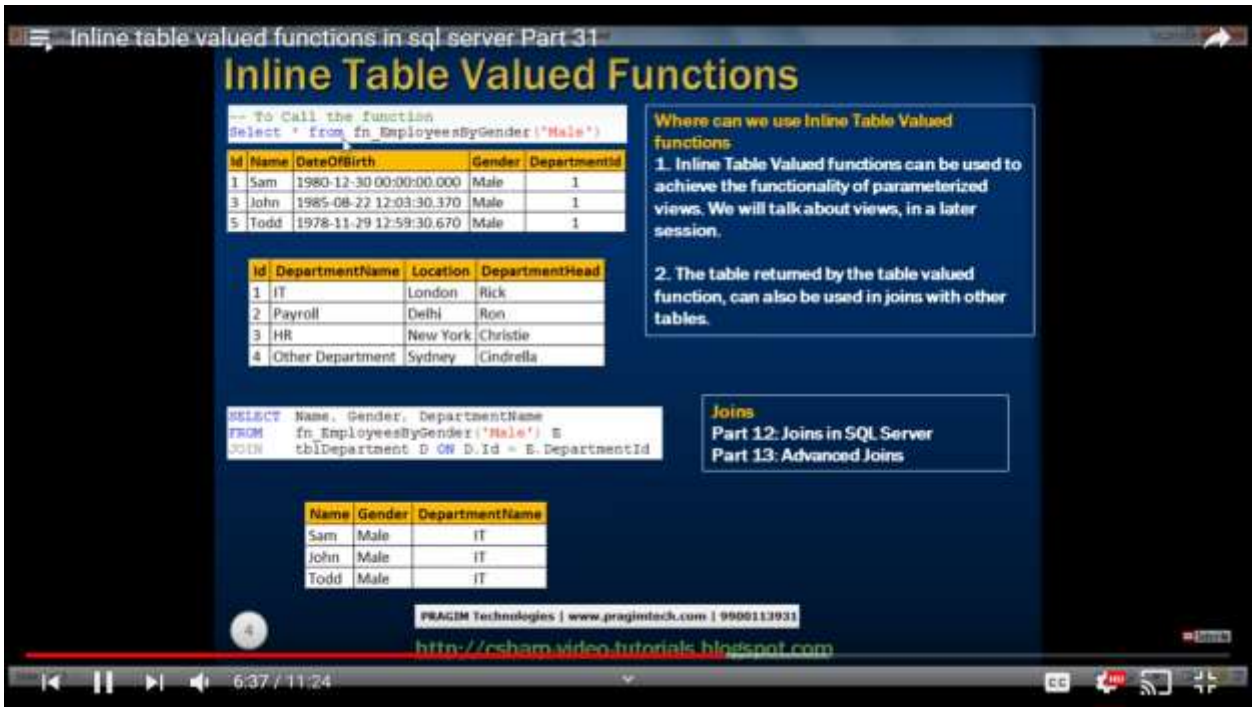

CREATE FUNCTION CalculateAge(@DOB Date)
RETURNS INT
AS
BEGIN
DECLARE @DOB DATE
DECLARE @Age INT
SET @DOB = '11/08/2011'
SET @Age = DATEDIFF(YEAR, @DOB, GETDATE()) -
CASE
WHEN (MONTH(@DOB) > MONTH(GETDATE())) OR
(MONTH(@DOB) = MONTH(GETDATE()) AND DAY(@DOB) > DAY(GETDATE()))
THEN 1
ELSE 0
END
SELECT @Age
END
Query executed successfully.


Scalar user defined functions in sql server Part 30
SCALAR UDF
CREATE FUNCTION Age(@DOB Date)
RETURNS INT
AS
BEGIN
DECLARE @Age INT
SET @Age = DATEDIFF(YEAR, @DOB, GETDATE()) -
CASE
WHEN (MONTH(@DOB) > MONTH(GETDATE())) OR
(MONTH(@DOB) = MONTH(GETDATE()) AND DAY(@DOB) > DAY(GETDATE()))
THEN 1
ELSE 0
END
RETURN @Age
END
When calling a scalar user-defined function, you must supply a two-part name, OwnerName.FunctionName. dbo stands for database owner.
Select dbo.Age('10/08/1982')
You can also invoke it using the complete 3 part name, DatabaseName.OwnerName.FunctionName
Select SampleDB.dbo.Age(dbo.Age('10/08/1982')
PRAGIM Technologies | www.pragimtech.com | 9900113931
4
http://csharp.video.tutorials.blogspot.com
12:32 / 19:06

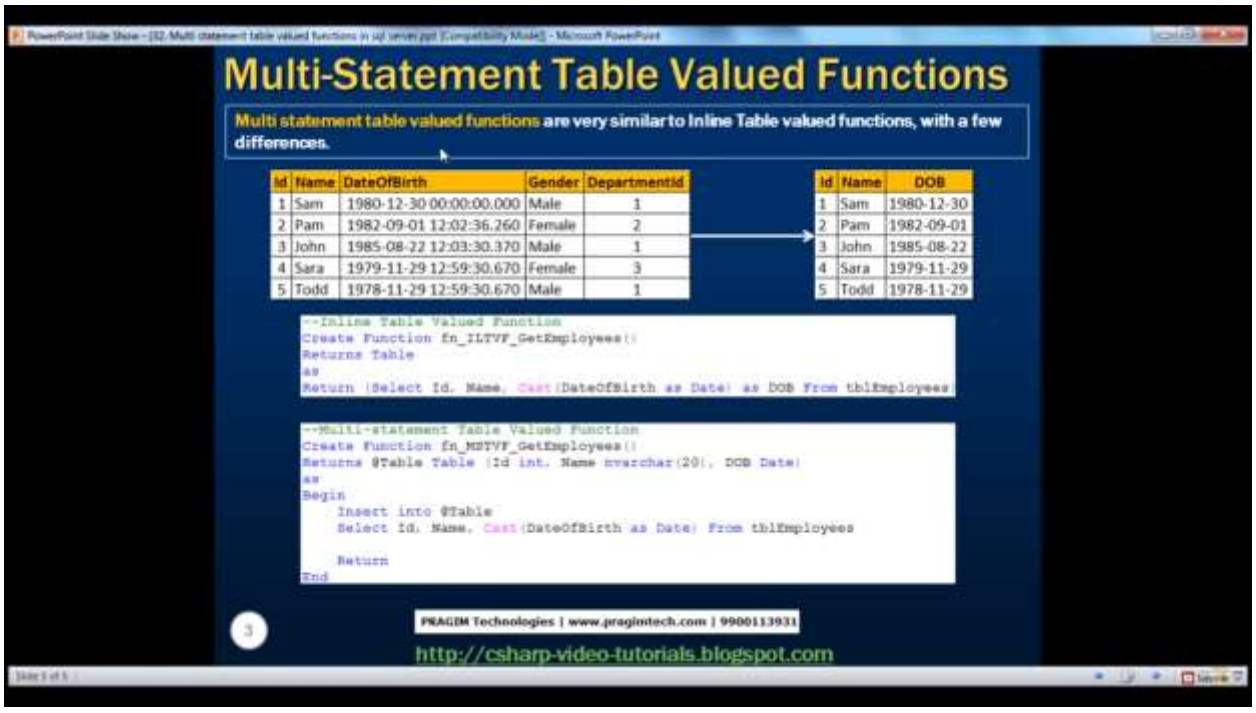

Scalar user defined functions in sql server Part 30
SCALAR UDF
--Scalar user defined functions can be used in the Select clause
Select Name, DateOfBirth, dbo.Age(DateOfBirth) as Age from tblEmployees
Id Name DateOfBirth
1 Sam 1980-12-30 00:00:00.000
2 Pam 1982-09-01 12:02:36.260
3 John 1985-08-22 12:03:30.370
4 Sara 1979-11-29 12:59:30.670
Name DateOfBirth Age
Sam 1980-12-30 00:00:00.000 31
Pam 1982-09-01 12:02:36.260 30
John 1985-08-22 12:03:30.370 27
Sara 1979-11-29 12:59:30.670 32
--Scalar user defined functions can be used in the Select Where clauses
Select Name, DateOfBirth, dbo.Age(DateOfBirth) as Age from tblEmployees
Where dbo.Age(DateOfBirth) > 30
Id Name DateOfBirth
1 Sam 1980-12-30 00:00:00.000
2 Pam 1982-09-01 12:02:36.260
3 John 1985-08-22 12:03:30.370
4 Sara 1979-11-29 12:59:30.670
Name DateOfBirth Age
Sam 1980-12-30 00:00:00.000 31
Sara 1979-11-29 12:59:30.670 32
A stored procedure also can accept DateOfBirth and return Age, but you cannot use stored procedures in a select or where clause. This is just one difference between a function and a stored procedure. There are several other differences, which we will talk about in a later session.
To alter a function we use ALTER FUNCTION FuncationName statement and to delete it, we use DROP FUNCTION FuncationName.
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
5
12:44 / 19:06

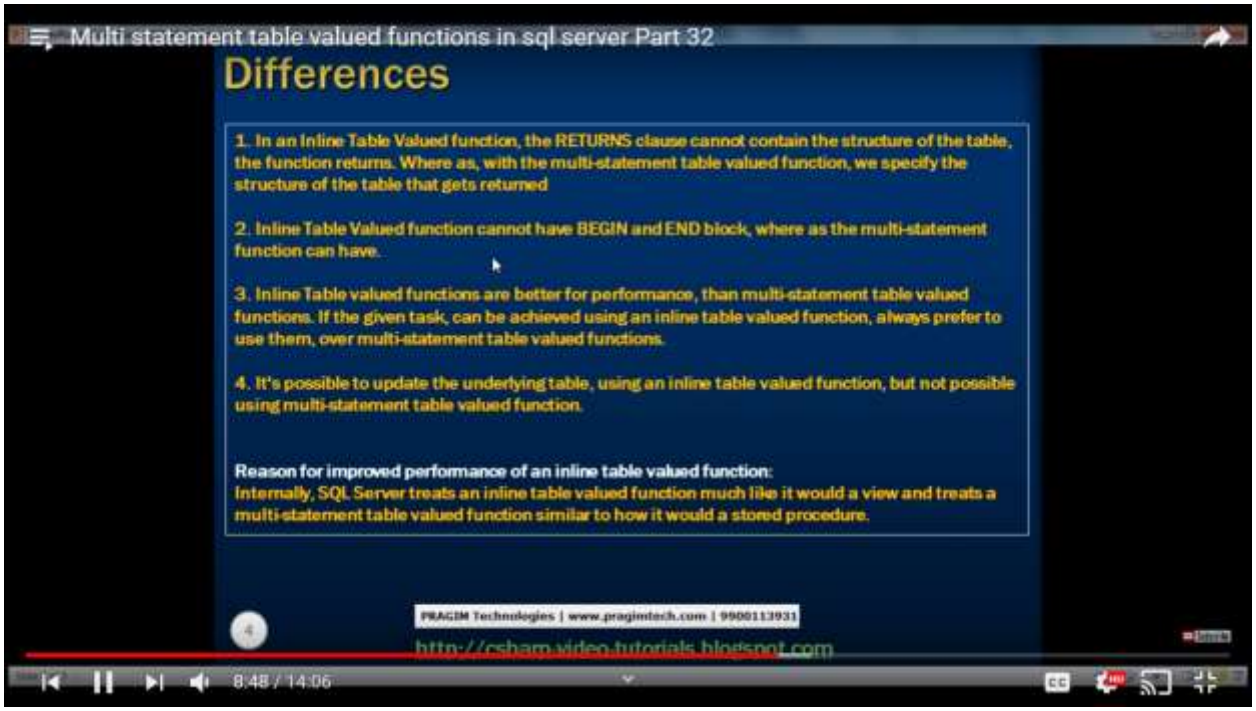

In this session we will learn
Creating Inline Table Valued Function
How to call an Inline Table Valued Function
Where do we use inline table valued functions
Pre-requisite
Part 30 – User Defined Functions
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
2

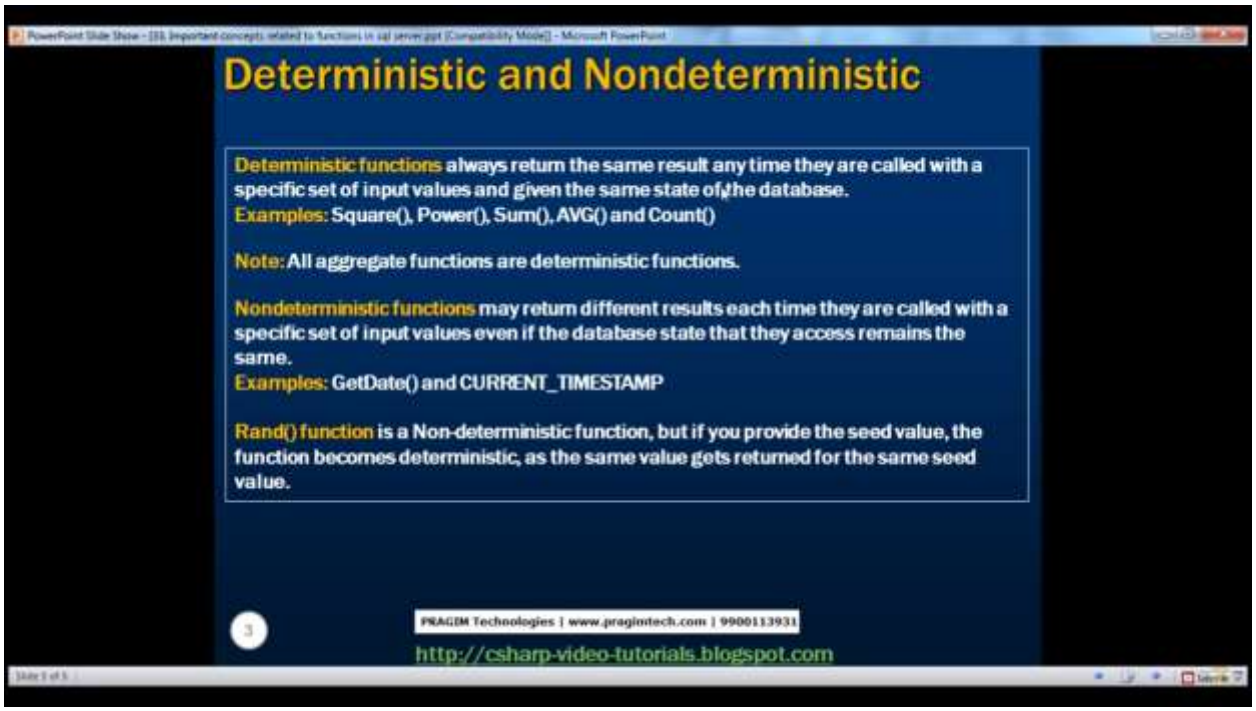

Inline table valued functions in sql server Part 31
Inline Table Valued Functions
In Part 30 of this video series we have seen how to create and call 'scalar user defined functions'. In this part, we will learn about 'Inline Table Valued Functions'
SCALAR FUNCTION – RETURNS A SCALAR VALUE
INLINE TABLE VALUED FUNCTION – RETURNS A TABLE
Id Name DateOfBirth Gender DepartmentId
1 Sam 1980-12-30 00:00:00.000 Male 1
2 Pam 1982-09-01 12:02:36.260 Female 2
3 John 1985-08-22 12:03:30.370 Male 1
4 Sara 1979-11-29 12:59:30.670 Female 3
5 Todd 1978-11-29 12:59:30.670 Male 1
Id Name DateOfBirth Gender DepartmentId
1 Sam 1980-12-30 00:00:00.000 Male 1
3 John 1985-08-22 12:03:30.370 Male 1
5 Todd 1978-11-29 12:59:30.670 Male 1
CREATE FUNCTION fn_EmployeesByGender(@Gender nvarchar(10))
RETURNS TABLE
AS
RETURN (Select Id, Name, DateOfBirth, Gender, DepartmentId
from tblEmployees
where Gender = @Gender)
1. We specify TABLE as the return type, instead of any scalar data type
2. The function body is not enclosed between BEGIN and END block.
3. The structure of the table that gets returned, is determined by the SELECT statement with in the function.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:53 / 11:24


Inline table valued functions in sql server Part 31
Inline Table Valued Functions
-- To Call the function
Select * from fn_EmployeesByGender('Male')
Id Name DateOfBirth Gender DepartmentId
1 Sam 1980-12-30 00:00:00.000 Male 1
3 John 1985-08-22 12:03:30.370 Male 1
5 Todd 1978-11-29 12:59:30.670 Male 1
Where can we use Inline Table Valued functions
1. Inline Table Valued functions can be used to achieve the functionality of parameterized views. We will talk about views, in a later session.
2. The table returned by the table valued function, can also be used in joins with other tables.
Id DepartmentName Location DepartmentHead
1 IT London Rick
2 Payroll Delhi Ron
3 HR New York Christie
4 Other Department Sydney Cindrella
SELECT Name, Gender, DepartmentName
FROM fn_EmployeesByGender('Male') E
JOIN tblDepartment D ON D.Id = E.DepartmentId
Joins
Part 12: Joins in SQL Server
Part 13: Advanced Joins
Name Gender DepartmentName
Sam Male IT
John Male IT
Todd Male IT
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
6:37 / 11:24


Multi statement table valued functions in sql server Part 32
In this session we will learn
Creating Multi-Statement Table Valued Function
Difference between inline and multi-statement Table Valued Functions
Pre-requisite
Part 30 – User Defined Functions
Part 31 – Inline Table Valued Functions
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
2
Settings
0:15 / 14:06


Multi-Statement Table Valued Functions
Multi statement table valued functions are very similar to Inline Table valued functions, with a few differences.
Id Name DateOfBirth Gender DepartmentId
1 Sam 1980-12-30 00:00:00.000 Male 1
2 Pam 1982-09-01 12:02:36.260 Female 2
3 John 1985-08-22 12:03:30.370 Male 1
4 Sara 1979-11-29 12:59:30.670 Female 3
5 Todd 1978-11-29 12:59:30.670 Male 1
Id Name DOB
1 Sam 1980-12-30
2 Pam 1982-09-01
3 John 1985-08-22
4 Sara 1979-11-29
5 Todd 1978-11-29
--Inline Table Valued Function
Create Function fn_ILTVF_GetEmployees()
Returns Table
as
Return (Select Id, Name, Cast(DateOfBirth as Date) as DOB from tblEmployees
--Multi-statement Table Valued Function
Create Function fn_MSTVF_GetEmployees()
Returns @Table Table (Id int, Name nvarchar(20), DOB Date)
as
Begin
Insert into @Table
Select Id, Name, Cast(DateOfBirth as Date) from tblEmployees
Return
End
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

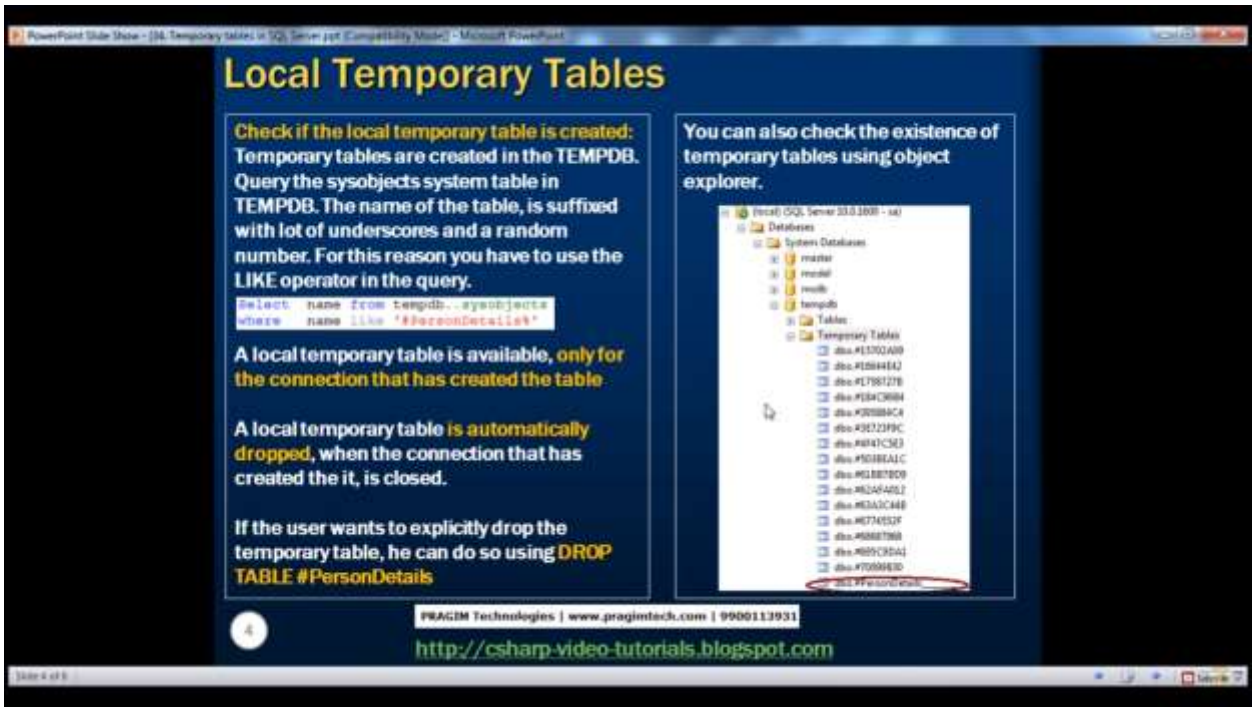

Multi statement table valued functions in sql server Part 32
Differences
1. In an Inline Table Valued function, the RETURNS clause cannot contain the structure of the table, the function returns. Where as, with the multi-statement table valued function, we specify the structure of the table that gets returned
2. Inline Table Valued function cannot have BEGIN and END block, where as the multi-statement function can have.
3. Inline Table valued functions are better for performance, than multi-statement table valued functions. If the given task, can be achieved using an inline table valued function, always prefer to use them, over multi-statement table valued functions.
4. It's possible to update the underlying table, using an inline table valued function, but not possible using multi-statement table valued function.
Reason for improved performance of an inline table valued function:
Internally, SQL Server treats an inline table valued function much like it would a view and treats a multi-statement table valued function similar to how it would a stored procedure.
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
8:48 / 14:06


Deterministic and Nondeterministic
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.
Examples: Square(), Power(), Sum(), AVG() and Count()
Note: All aggregate functions are deterministic functions.
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.
Examples: GetDate() and CURRENT_TIMESTAMP
Rand() function is a Non-deterministic function, but if you provide the seed value, the function becomes deterministic, as the same value gets returned for the same seed value.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

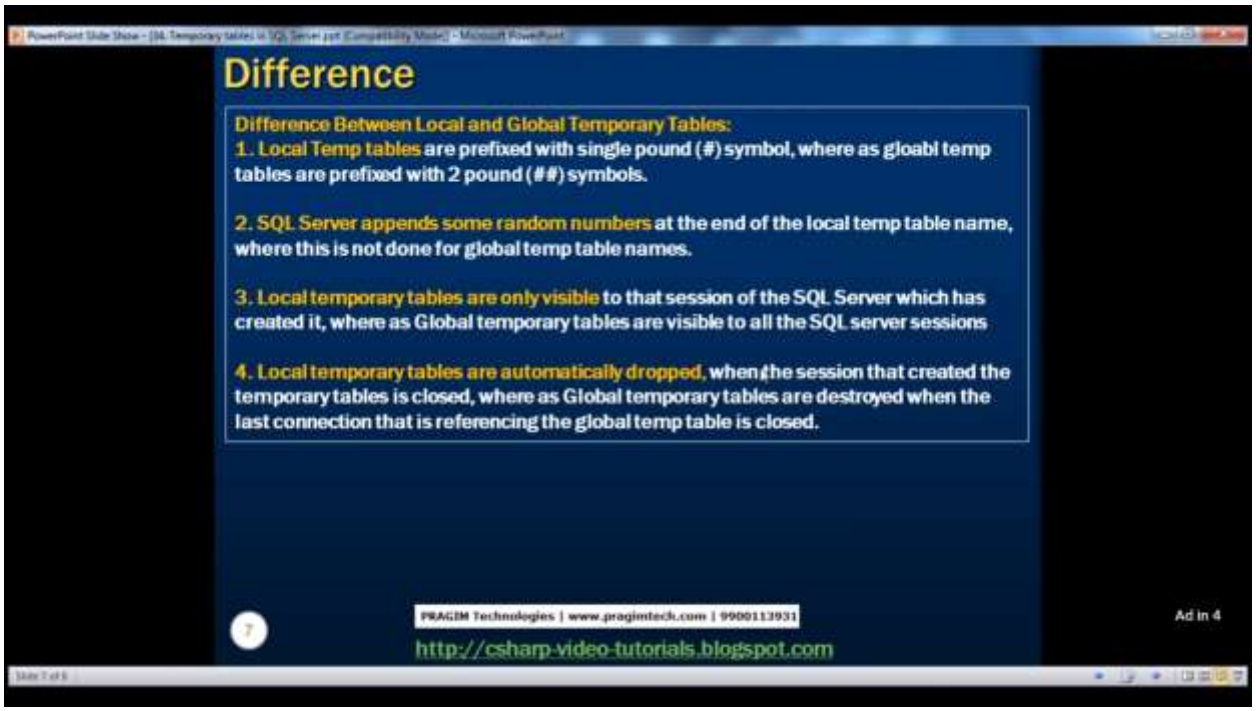

WITH ENCRYPTION and SCHEMABINDING
Encrypting a function definiton using WITH ENCRYPTION OPTION:
We have learnt how to encrypt Stored procedure text using WITH ENCRYPTION OPTION in Part 18 of this video series. Along the same lines, you can also encrypt a function text. Once, encrypted, you cannot view the text of the function, using sp_helptext system stored procedure. If you try to, you will get a message stating 'The text for object is encrypted.' There are ways to decrypt, which is beyond the scope of this video.
Use WITH ENCRYPTION
Creating a function WITH SCHEMABINDING option:
Schemabinding, specifies that the function is bound to the database objects that it references. When SCHEMABINDING is specified, the base objects cannot be modified in any way that would affect the function definition. The function definition itself must first be modified or dropped to remove dependencies on the object that is to be modified.
Use WITH SCHEMABINDING
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com


In this session we will learn
What are temporary tables
Types of temporary tables – Local and Global
Difference between local and global temp tables
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
2


Temporary tables in SQL Server Part 34
Temporary Tables
What are Temporary tables?
Temporary tables, are very similar to the permanent tables. Permanent tables get created in the database you specify, and remain in the database permanently, until you delete (drop) them. On the other hand, temporary tables get created in the TempDB and are automatically deleted, when they are no longer used.
Different Types of Temporary tables
1. Local Temporary Tables
2. Global Temporary tables.
--Create Local Temporary Table
Create Table #PersonDetails(Id int, Name nvarchar(20))
--Insert Data into the temporary table
Insert into #PersonDetails Values(1, 'Mike')
Insert into #PersonDetails Values(2, 'John')
Insert into #PersonDetails Values(3, 'Todd')
--Select data from the temporary table
Select * from #PersonDetails
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:22 / 15:16

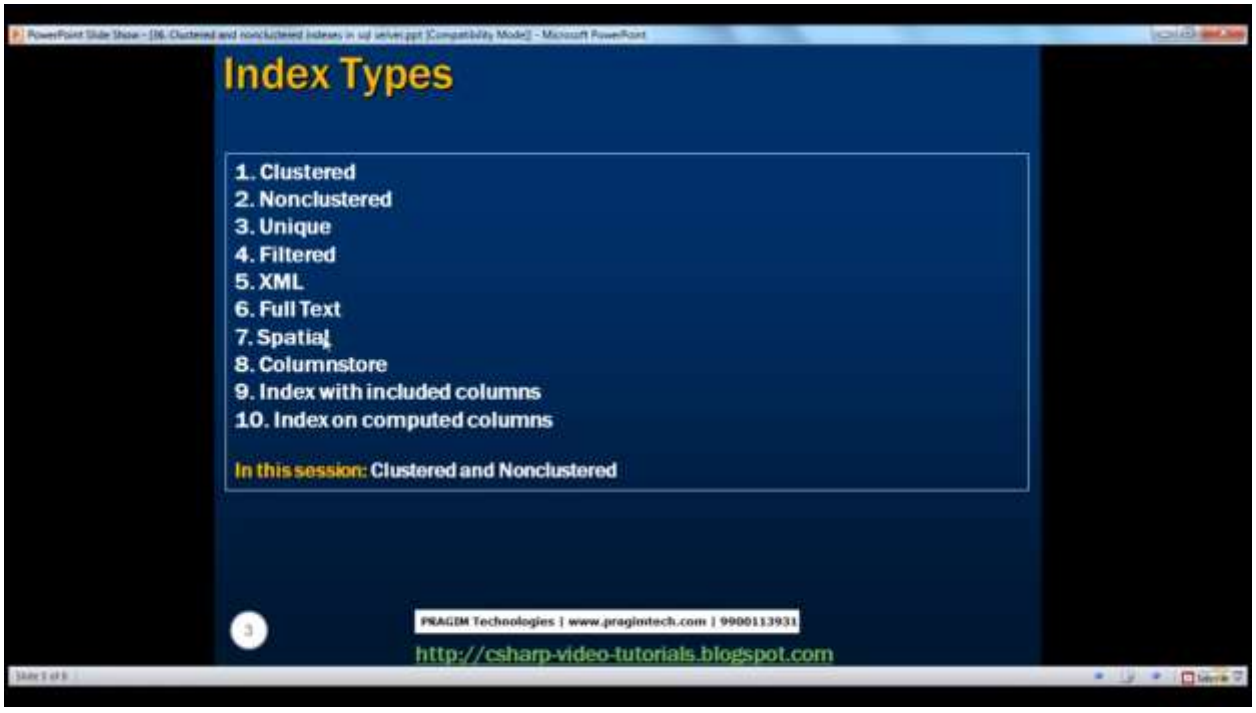

Local Temporary Tables
Check if the local temporary table is created:
Temporary tables are created in the TEMPDB. Query the sysobjects system table in TEMPDB. The name of the table, is suffixed with lot of underscores and a random number. For this reason you have to use the LIKE operator in the query.
Select name from tempdb..sysobjects
where name like '#PersonDetails%'
A local temporary table is available, only for the connection that has created the table
A local temporary table is automatically dropped, when the connection that has created the it, is closed.
If the user wants to explicitly drop the temporary table, he can do so using DROP TABLE #PersonDetails
You can also check the existence of temporary tables using object explorer.
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com


Global Temporary Tables
To create a Global Temporary Table, prefix the name of the table with 2 pound (##) symbols.
Create Table ##EmployeeDetails(Id int, Name nvarchar(20))
Global temporary tables are visible to all the connections of the sql server, and are only destroyed when the last connection referencing the table is closed.
Multiple users, across multiple connections can have local temporary tables with the same name, but, a global temporary table name has to be unique, and if you inspect the name of the global temp table, in the object explorer, there will be no random numbers suffixed at the end of the table name.
6
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com


Difference
Difference Between Local and Global Temporary Tables:
1. Local Temp tables are prefixed with single pound (#) symbol, where as gloabl temp tables are prefixed with 2 pound (##) symbols.
2. SQL Server appends some random numbers at the end of the local temp table name, where this is not done for global temp table names.
3. Local temporary tables are only visible to that session of the SQL Server which has created it, where as Global temporary tables are visible to all the SQL server sessions
4. Local temporary tables are automatically dropped, when the session that created the temporary tables is closed, where as Global temporary tables are destroyed when the last connection that is referencing the global temp table is closed.
7
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
Ad in 4

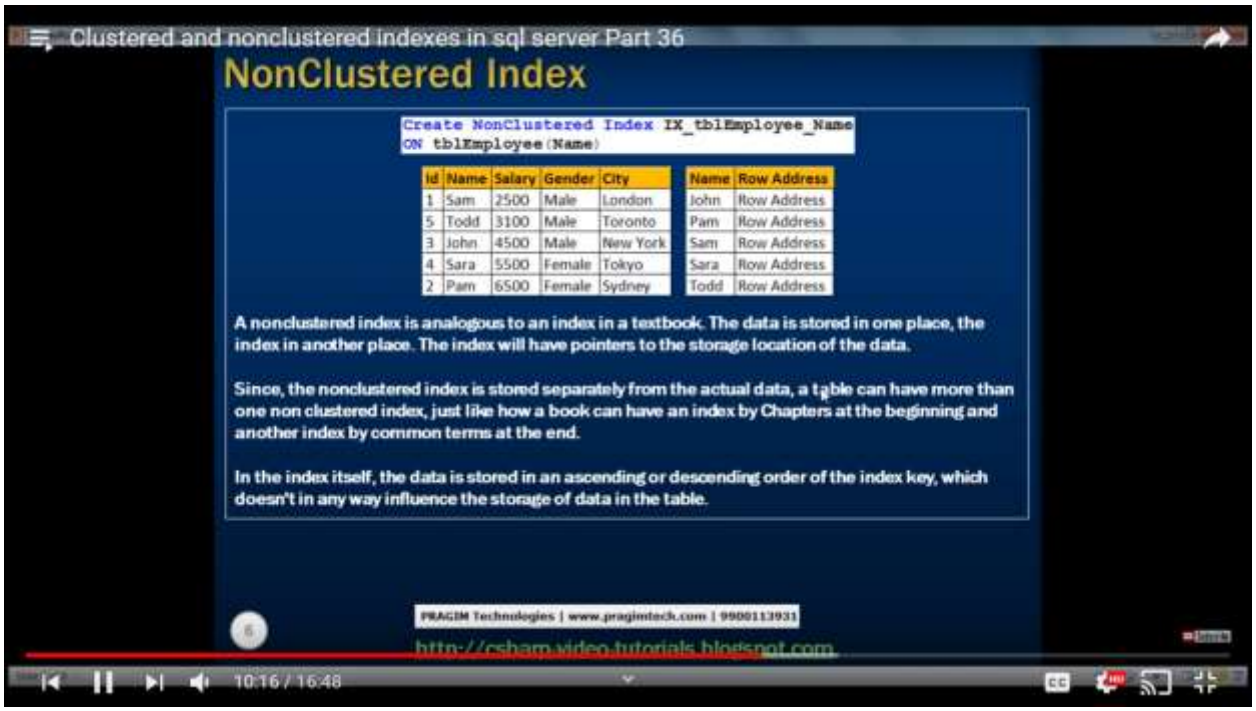

Why Indexes
Indexes are used by queries to find data from tables quickly. Indexes are created on tables and views. Index on a table or a view, is very similar to an index that we find in a book.
If you don't have an index, and I ask you to locate a specific chapter in the book, you will have to look at every page starting from the first page of the book.
On, the other hand, if you have the index, you lookup the page number of the chapter in the index, and then directly go to that page number to locate the chapter.
Obviously, the book index is helping to drastically reduce the time it takes to find the chapter.
In a similar way, Table and View indexes, can help the query to find data quickly.
In fact, the existence of the right indexes, can drastically improve the performance of the query. If there is no index to help the query, then the query engine, checks every row in the table from the beginning to the end. This is called as Table Scan. Table scan is bad for performance.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

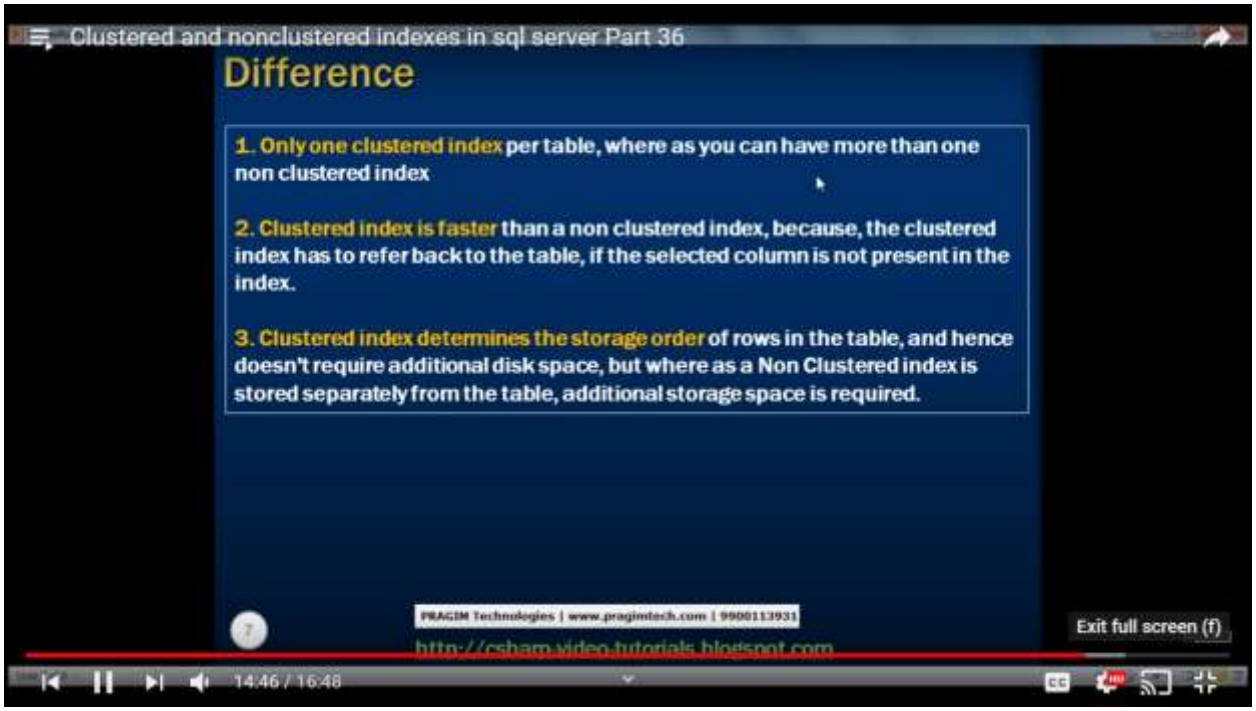

Creating an Index
CREATE Index IX_tblEmployee_Salary
ON tblEmployee (SALARY ASC)
The index stores salary of each employee, in the ascending order as shown below. The actual index may look slightly different.
Id Name Salary Gender
1 Sam 2500 Male
2 Pam 6500 Female
3 John 4500 Male
4 Sara 5500 Female
5 Todd 3100 Male
Salary RowAddress
2500 Row Address
3100 Row Address
4500 Row Address
5500 Row Address
6500 Row Address
Now, when the SQL server has to execute the same query, it has an index on the salary column to help this query. Salaries between the range of 5000 and 7000 are usually present at the bottom, since the salaries are arranged in an ascending order. SQL server picks up the row addresses from the index and directly fetch the records from the table, rather than scanning each row in the table. This is called as Index Seek.
Next Video – Different Types of indexes, Disadvantage etc...
5
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

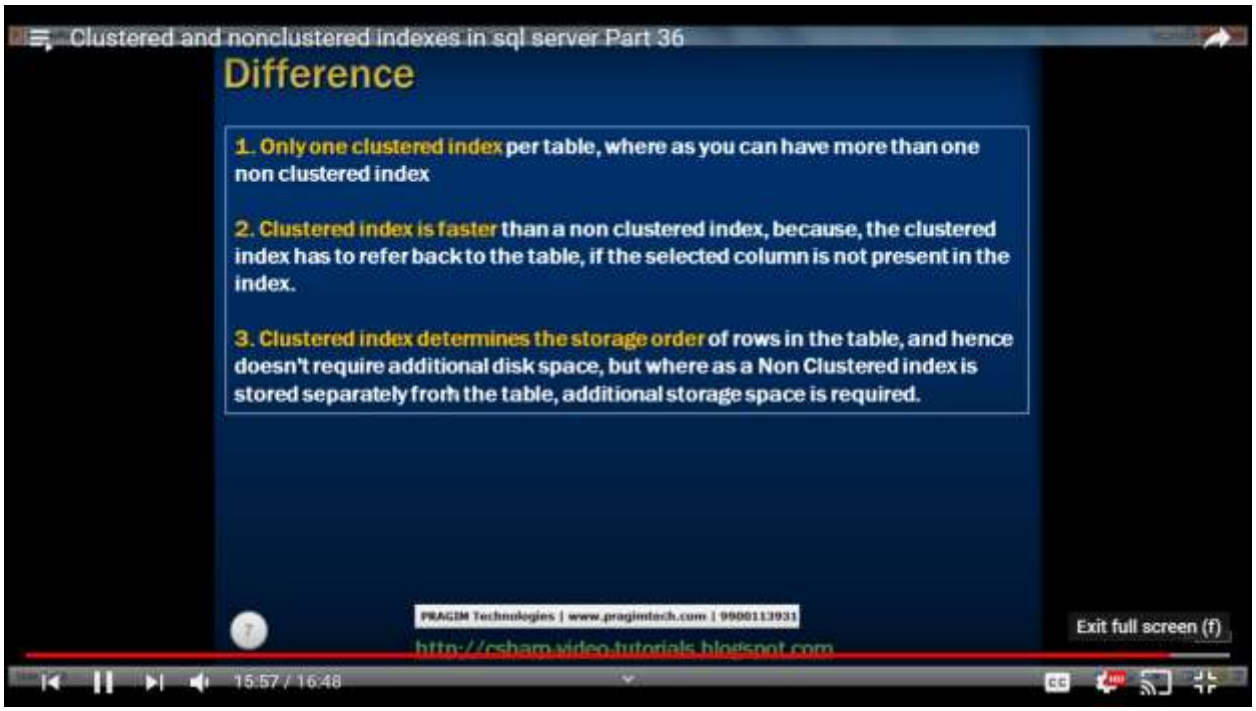

Index Types
1. Clustered
2. Nonclustered
3. Unique
4. Filtered
5. XML
6. Full Text
7. Spatial
8. Columnstore
9. Index with included columns
10. Index on computed columns
In this session: Clustered and Nonclustered
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

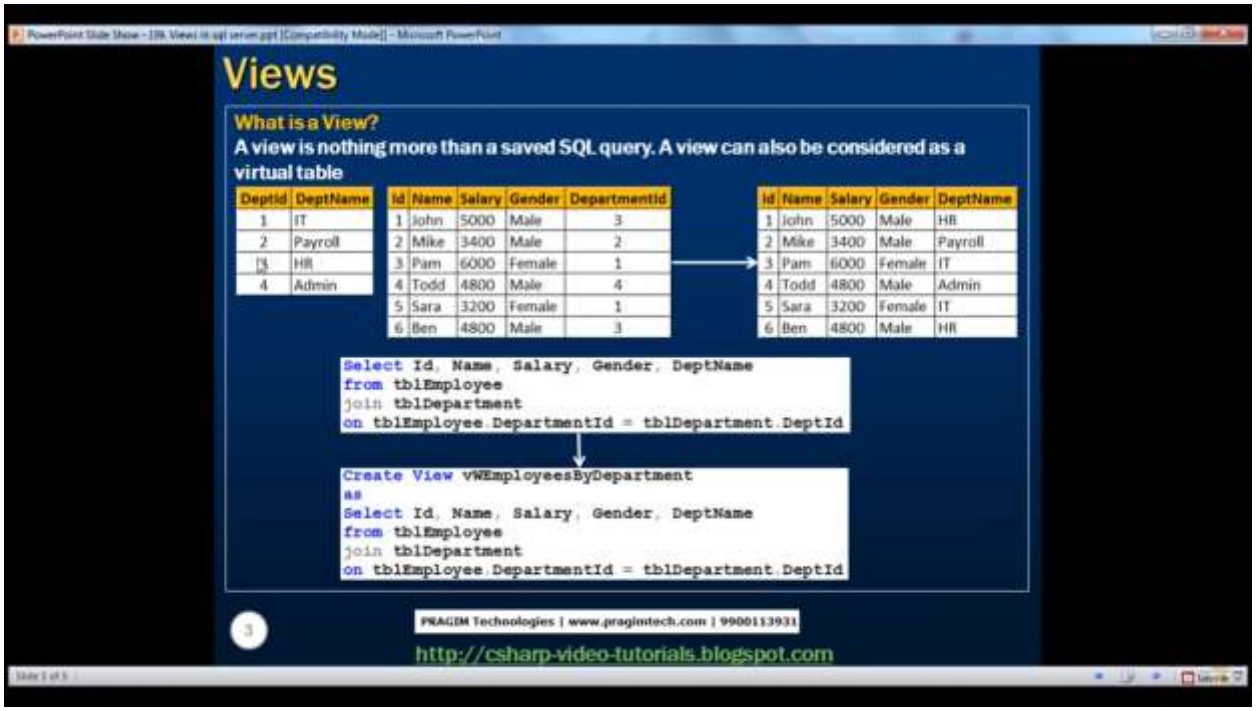

Clustered Index
A clustered index determines the physical order of data in a table. For this reason, a table can have only one clustered index.
CREATE TABLE [tblEmployee]
(
[Id] int Primary Key,
[Name] nvarchar(50),
[Salary] int,
[Gender] nvarchar(10),
[City] nvarchar(50)
)
Note that Id column is marked as primary key. Primary key, constraint create clustered indexes automatically if no clustered index already exists on the table.
To confirm: Execute sp_helpindex tblEmployee
Note that, the values for Id column are not in a sequential order
Insert into tblEmployee Values(3,'John',4500,'Male','New York')
Insert into tblEmployee Values(1,'Sam',2500,'Male','London')
Insert into tblEmployee Values(4,'Sara',5500,'Female','Tokyo')
Insert into tblEmployee Values(5,'Todd',3100,'Male','Toronto')
Insert into tblEmployee Values(2,'Pam',6500,'Female','Sydney')
Select * from tblEmployee
Id Name Salary Gender City
1 Sam 2500 Male London
2 Pam 6500 Female Sydney
3 John 4500 Male New York
4 Sara 5500 Female Tokyo
5 Todd 3100 Male Toronto
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

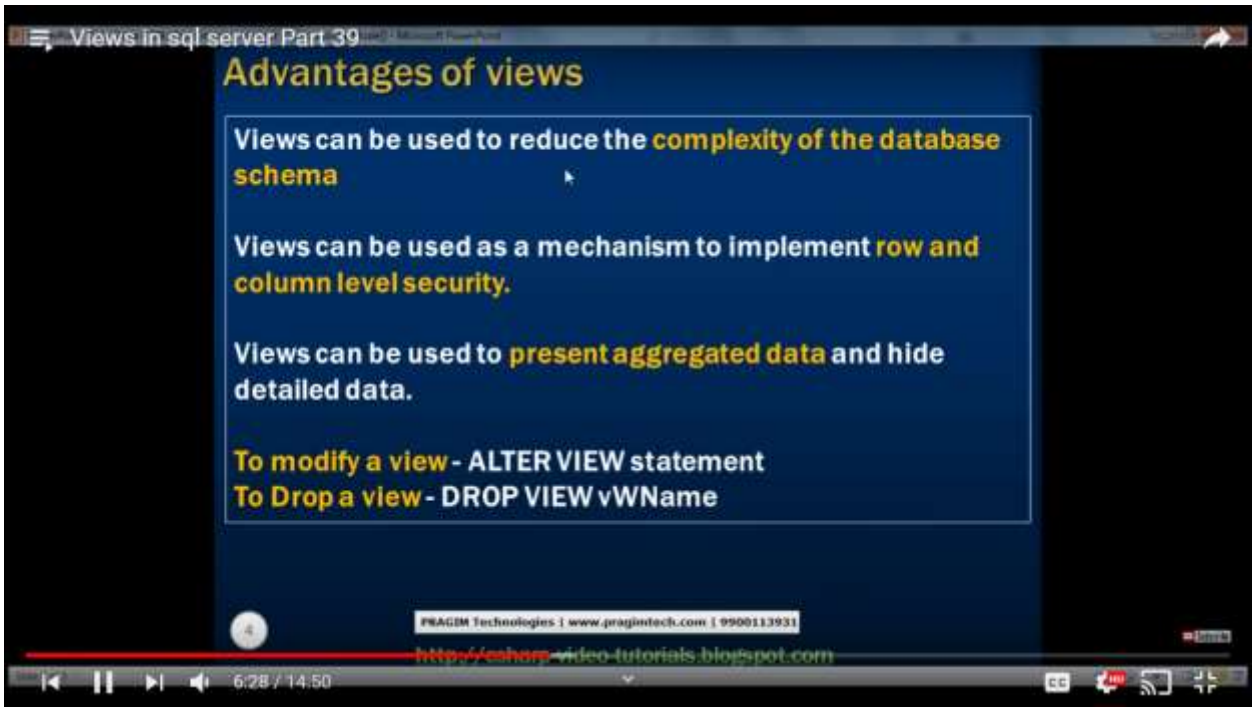

Clustered and nonclustered indexes in sql server Part 36
Clustered Index
A clustered index is analogous to a telephone directory, where the data is arranged by the last name. We just learnt that, a table can have only one clustered index. However, the index can contain multiple columns (a composite index), like the way a telephone directory is organized by last name and first name.
Create a composite clustered Index on the Gender and Salary columns
Create Clustered Index IX_tblEmployee_Gender_Salary
ON tblEmployee(Gender DESC, Salary ASC)
Select * from tblEmployee
Id Name Salary Gender City
1 Sam 2500 Male London
5 Todd 3100 Male Toronto
3 John 4500 Male New York
4 Sara 5500 Female Tokyo
2 Pam 6500 Female Sydney
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
5:22 / 16:48

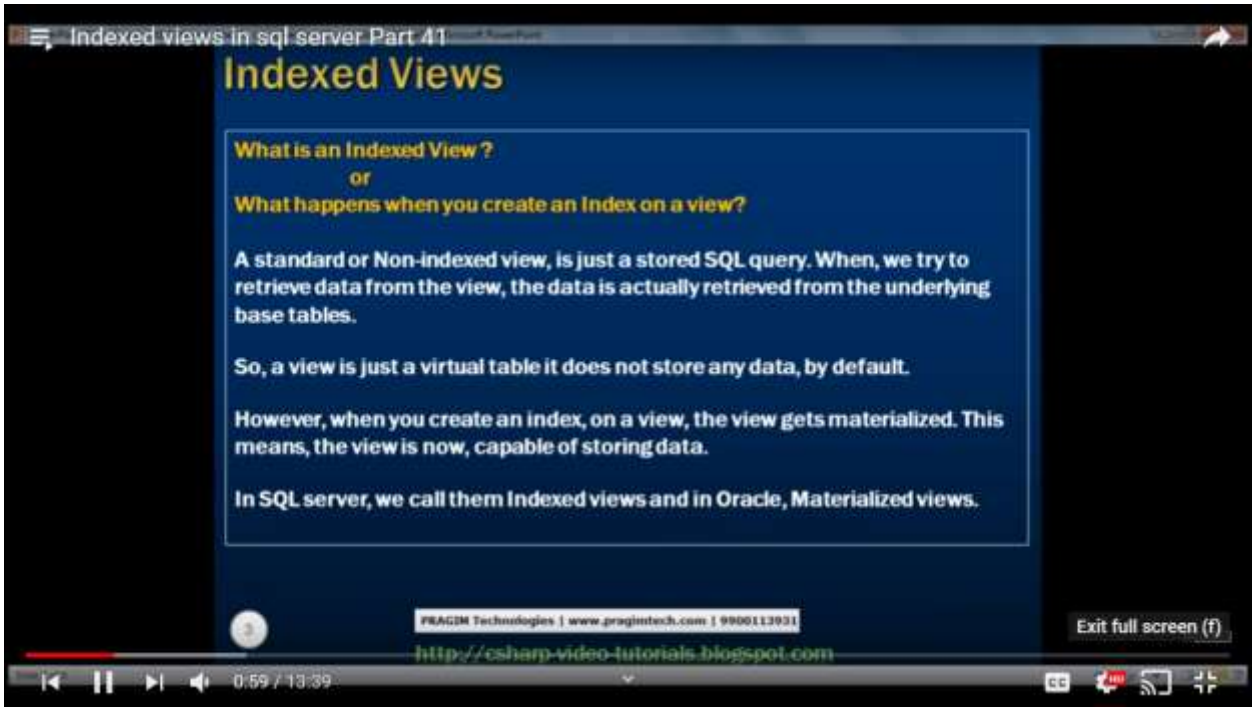

Clustered and nonclustered indexes in sql server Part 36
NonClustered Index
Create NonClustered Index IX_tblEmployee_Name
ON tblEmployee(Name)
Id Name Salary Gender City
1 Sam 2500 Male London
5 Todd 3100 Male Toronto
3 John 4500 Male New York
4 Sara 5500 Female Tokyo
2 Pam 6500 Female Sydney
Name Row Address
John Row Address
Pam Row Address
Sam Row Address
Sara Row Address
Todd Row Address
A nonclustered index is analogous to an index in a textbook. The data is stored in one place, the index in another place. The index will have pointers to the storage location of the data.
Since, the nonclustered index is stored separately from the actual data, a table can have more than one non clustered index, just like how a book can have an index by Chapters at the beginning and another index by common terms at the end.
In the index itself, the data is stored in an ascending or descending order of the index key, which doesn't in any way influence the storage of data in the table.
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
10:16 / 16:48

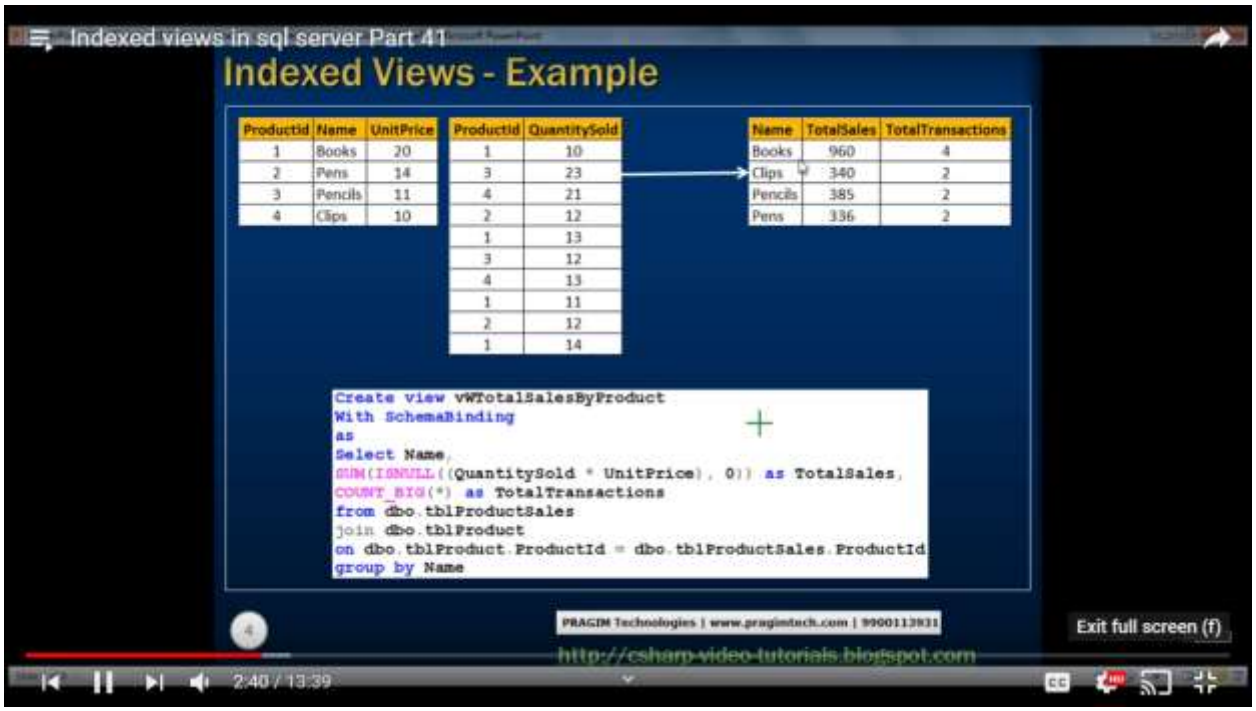

Clustered and nonclustered indexes in sql server Part 36
Difference
1. Only one clustered index per table, where as you can have more than one non clustered index
2. Clustered index is faster than a non clustered index, because, the clustered index has to refer back to the table, if the selected column is not present in the index.
3. Clustered index determines the storage order of rows in the table, and hence doesn't require additional disk space, but where as a Non Clustered index is stored separately from the table, additional storage space is required.
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp.video.tutorials.blogspot.com
Exit full screen (f)
14:46 / 16:48

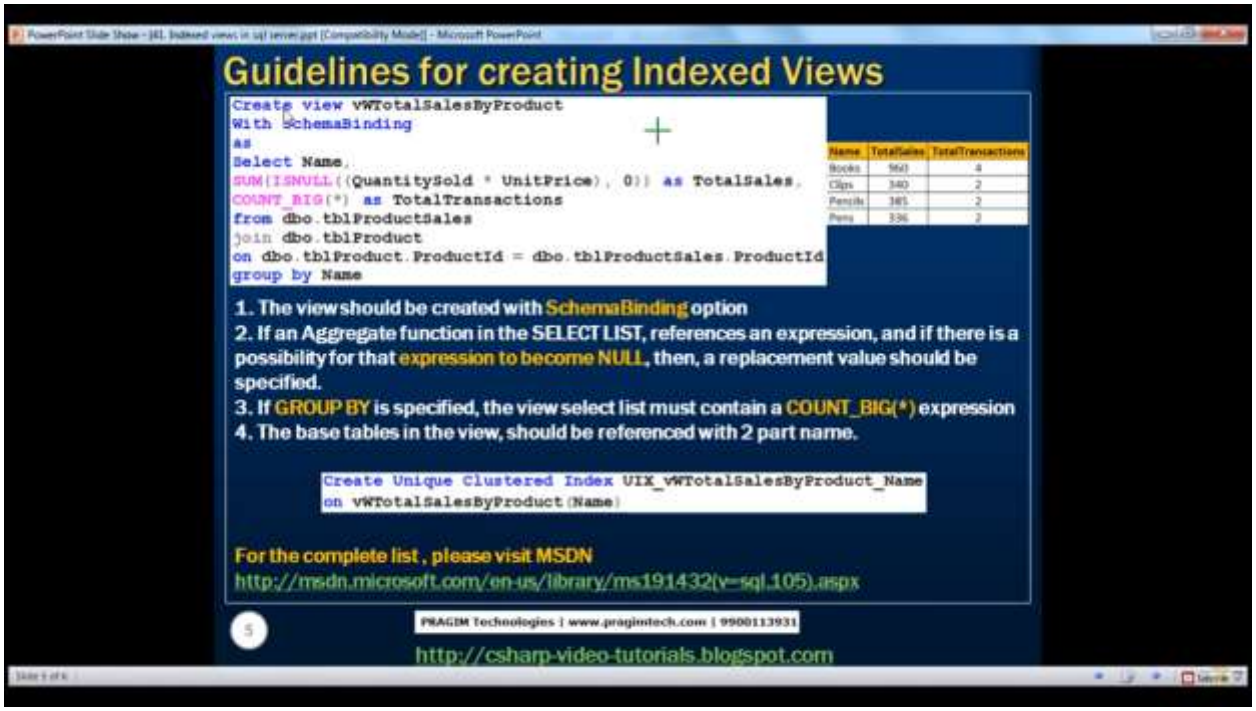

Clustered and nonclustered indexes in sql server Part 36
Difference
1. Only one clustered index per table, where as you can have more than one non clustered index
2. Clustered index is faster than a non clustered index, because, the clustered index has to refer back to the table, if the selected column is not present in the index.
3. Clustered index determines the storage order of rows in the table, and hence doesn't require additional disk space, but where as a Non Clustered index is stored separately from the table, additional storage space is required.
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp.video.tutorials.blogspot.com
Exit full screen (f)
15:57 / 16:48

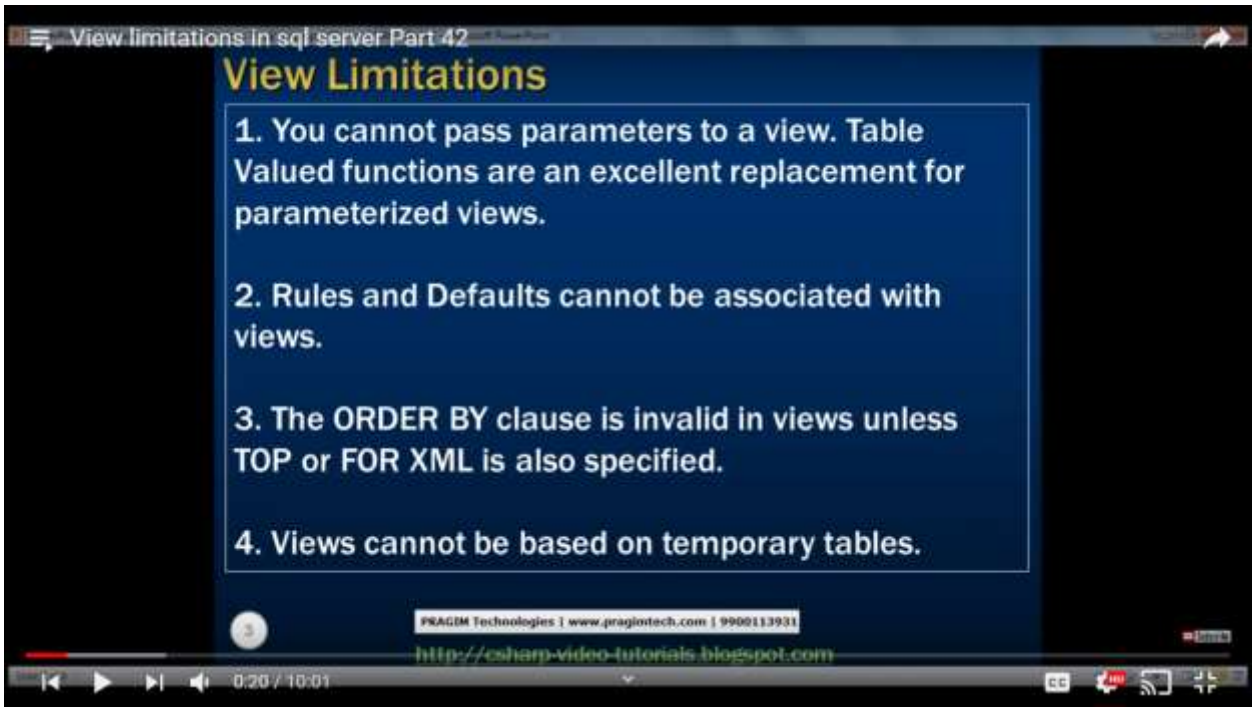

Views
What is a View?
A view is nothing more than a saved SQL query. A view can also be considered as a virtual table
DeptId DeptName
1 IT
2 Payroll
3 HR
4 Admin
Id Name Salary Gender DepartmentId
1 John 5000 Male 3
2 Mike 3400 Male 2
3 Pam 6000 Female 1
4 Todd 4800 Male 4
5 Sara 3200 Female 1
6 Ben 4800 Male 3
Id Name Salary Gender DeptName
1 John 5000 Male HR
2 Mike 3400 Male Payroll
3 Pam 6000 Female IT
4 Todd 4800 Male Admin
5 Sara 3200 Female IT
6 Ben 4800 Male HR
Select Id, Name, Salary, Gender, DeptName
from tblEmployee
join tblDepartment
on tblEmployee.DepartmentId = tblDepartment.DeptId
Create View vWEmployeesByDepartment
as
Select Id, Name, Salary, Gender, DeptName
from tblEmployee
join tblDepartment
on tblEmployee.DepartmentId = tblDepartment.DeptId
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

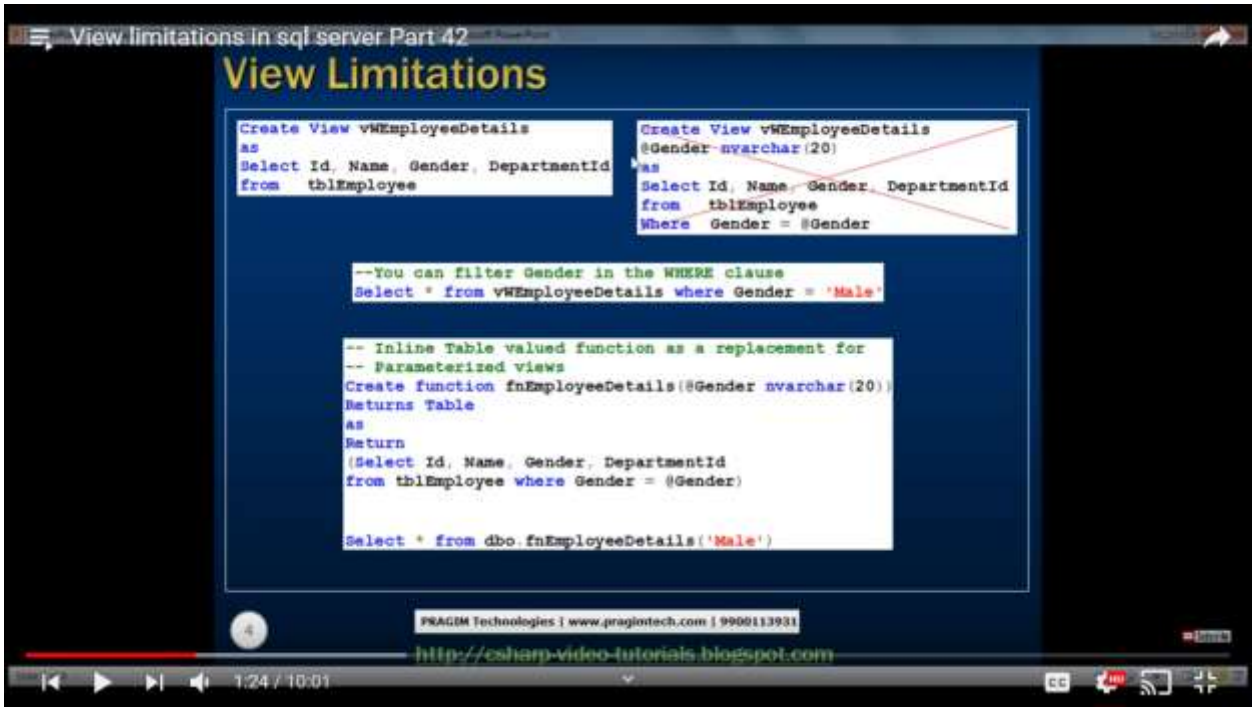

Views in sql server Part 39
Advantages of views
Views can be used to reduce the complexity of the database schema
Views can be used as a mechanism to implement row and column level security.
Views can be used to present aggregated data and hide detailed data.
To modify a view - ALTER VIEW statement
To Drop a view - DROP VIEW vWName
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
6:28 / 14:50

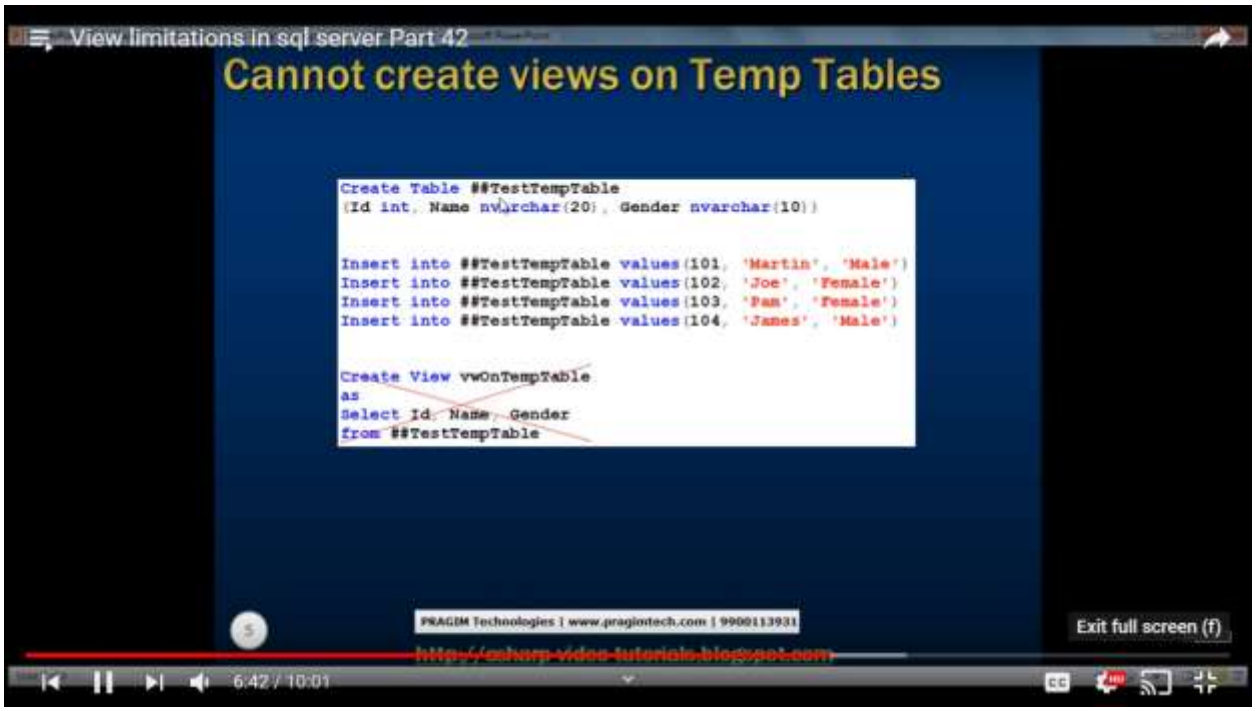

Indexed views in sql server Part 41
Indexed Views
What is an Indexed View ?
or
What happens when you create an Index on a view?
A standard or Non-indexed view, is just a stored SQL query. When, we try to retrieve data from the view, the data is actually retrieved from the underlying base tables.
So, a view is just a virtual table it does not store any data, by default.
However, when you create an index, on a view, the view gets materialized. This means, the view is now, capable of storing data.
In SQL server, we call them Indexed views and in Oracle, Materialized views.
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
Exit full screen (f)
0:59 / 13:39


Indexed views in sql server Part 41
Indexed Views - Example
Productid Name UnitPrice
1 Books 20
2 Pens 14
3 Pencils 11
4 Clips 10
Productid QuantitySold
1 10
3 23
4 21
2 12
1 13
3 12
4 13
1 11
2 12
1 14
Name TotalSales TotalTransactions
Books 960 4
Clips 340 2
Pencils 385 2
Pens 336 2
Create view vWTotalSalesByProduct
With SchemaBinding
as
Select Name,
SUM(ISNULL((QuantitySold * UnitPrice), 0)) as TotalSales,
COUNT_BIG(*) as TotalTransactions
from dbo.tblProductSales
join dbo.tblProduct
on dbo.tblProduct.ProductId = dbo.tblProductSales.ProductId
group by Name
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
Exit full screen (f)
2:40 / 13:39


Guidelines for creating Indexed Views
Create view vWTotalSalesByProduct
With SchemaBinding
as
Select Name,
SUM(ISNULL((QuantitySold * UnitPrice), 0)) as TotalSales,
COUNT_BIG(*) as TotalTransactions
from dbo.tblProductSales
join dbo.tblProduct
on dbo.tblProduct.ProductId = dbo.tblProductSales.ProductId
group by Name
Name TotalSales TotalTransactions
Books 960 4
Clips 340 2
Pencils 385 2
Pens 336 2
1. The view should be created with SchemaBinding option
2. If an Aggregate function in the SELECT LIST, references an expression, and if there is a possibility for that expression to become NULL, then, a replacement value should be specified.
3. If GROUP BY is specified, the view select list must contain a COUNT_BIG(*) expression
4. The base tables in the view, should be referenced with 2 part name.
Create Unique Clustered Index UIX_vWTotalSalesByProduct_Name
on vWTotalSalesByProduct(Name)
For the complete list , please visit MSDN
http://msdn.microsoft.com/en-us/library/ms191432(v=sql.105).aspx
5
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com


View limitations in sql server Part 42
View Limitations
1. You cannot pass parameters to a view. Table Valued functions are an excellent replacement for parameterized views.
2. Rules and Defaults cannot be associated with views.
3. The ORDER BY clause is invalid in views unless TOP or FOR XML is also specified.
4. Views cannot be based on temporary tables.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:20 / 10:01


View limitations in sql server Part 42
View Limitations
Create View vWEmployeeDetails
as
Select Id, Name, Gender, DepartmentId
from tblEmployee
Create View vWEmployeeDetails
@Gender nvarchar(20)
as
Select Id, Name, Gender, DepartmentId
from tblEmployee
Where Gender = @Gender
--You can filter Gender in the WHERE clause
Select * from vWEmployeeDetails where Gender = 'Male'
-- Inline Table valued function as a replacement for
-- Parameterized views
Create function fnEmployeeDetails(@Gender nvarchar(20))
Returns Table
as
Return
(Select Id, Name, Gender, DepartmentId
from tblEmployee where Gender = @Gender)
Select * from dbo.fnEmployeeDetails('Male')
PRAGIM Technologies | www.pragimtech.com | 9900113931
4
http://csharp-video-tutorials.blogspot.com
1:24 / 10:01


View limitations in sql server Part 42
Cannot create views on Temp Tables
Create Table ##TestTempTable
(Id int, Name nvarchar(20), Gender nvarchar(10))
Insert into ##TestTempTable values(101, 'Martin', 'Male')
Insert into ##TestTempTable values(102, 'Joe', 'Female')
Insert into ##TestTempTable values(103, 'Pam', 'Female')
Insert into ##TestTempTable values(104, 'James', 'Male')
Create View vwOnTempTable
as
Select Id, Name, Gender
from ##TestTempTable
PRAGIM Technologies | www.pragimtech.com | 9900113931
5
http://csharp-video-tutorials.blogspot.com
Exit full screen (f)
6:42 / 10:01

Triggers

In SQL server there are 3 types of triggers

1. DML triggers
2. DDL triggers
3. Logon trigger

DML triggers are fired automatically in response to DML events (INSERT, UPDATE & DELETE)

DML triggers can be again classified into 2 types

1. After triggers (Sometimes called as FOR triggers)
2. Instead of triggers

After triggers, fires after the triggering action. The INSERT, UPDATE, and DELETE statements, causes an after trigger to fire after the respective statements complete execution.

INSTEAD of triggers, fires instead of the triggering action. The INSERT, UPDATE, and DELETE statements, causes an INSTEAD OF trigger to fire INSTEAD OF the respective statement execution.

After Triggers

Id	Name	Salary	Gender	DepartmentId
1	John	5000	Male	3
2	Mike	3400	Male	2
3	Pam	6000	Female	1
4	Todd	4800	Male	4
5	Sara	3200	Female	1
6	Ben	4800	Male	3

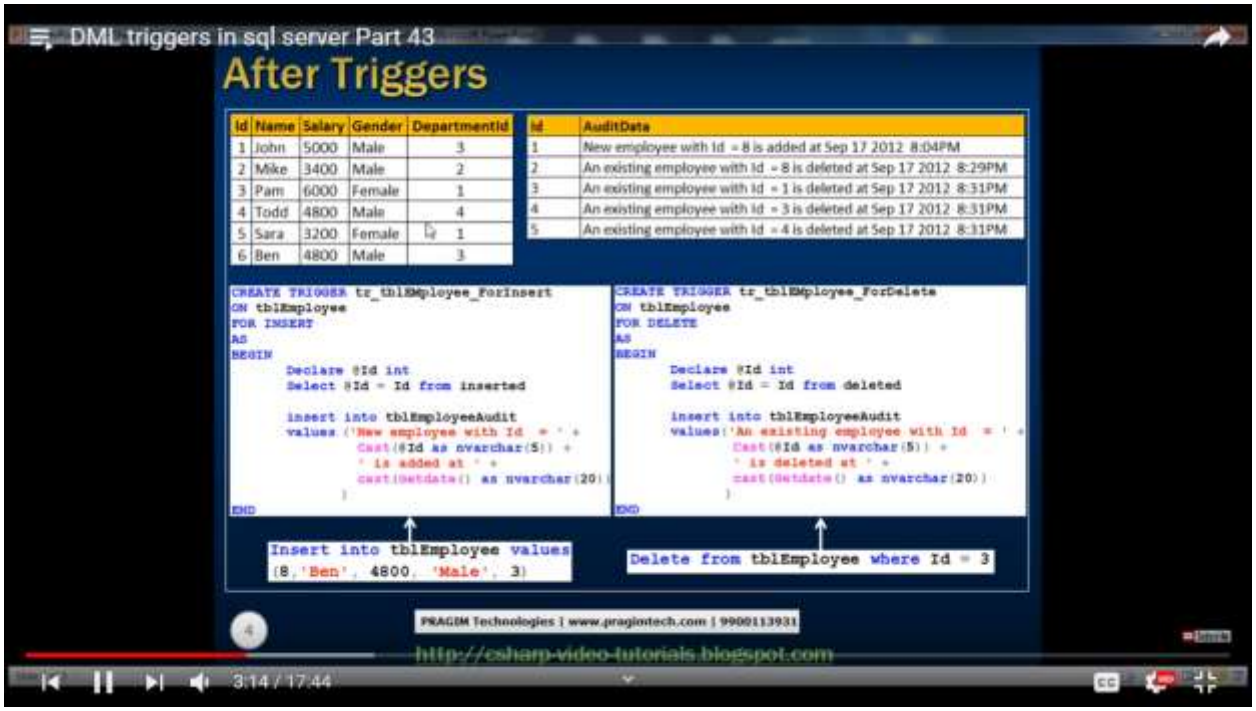
Id	AuditData
1	New employee with Id = 8 is added at Sep 17 2012 8:04PM
2	An existing employee with Id = 8 is deleted at Sep 17 2012 8:29PM
3	An existing employee with Id = 1 is deleted at Sep 17 2012 8:31PM
4	An existing employee with Id = 3 is deleted at Sep 17 2012 8:31PM
5	An existing employee with Id = 4 is deleted at Sep 17 2012 8:31PM

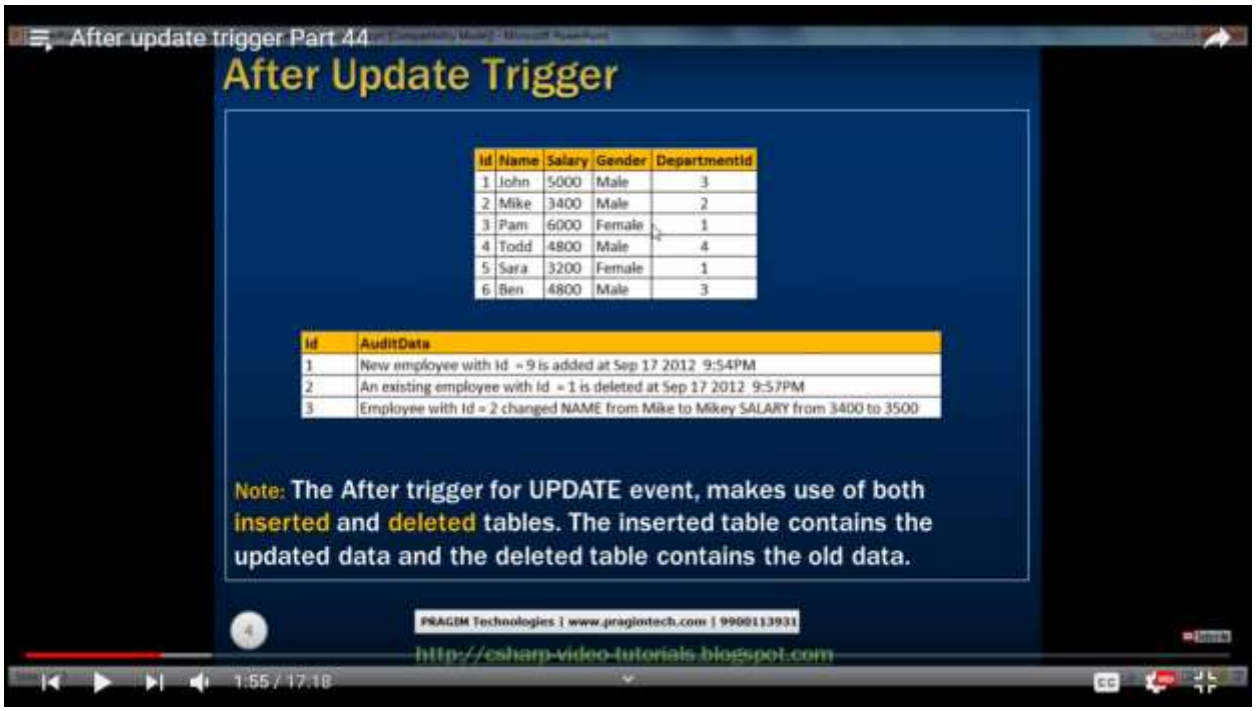
```
CREATE TRIGGER tr_tblEmployee_ForInsert
ON tblEmployee
FOR INSERT
AS
BEGIN
        Declare @Id int
        Select @Id = Id from inserted

        insert into tblEmployeeAudit
        values ('New employee with Id = ' +
                Cast(@Id as nvarchar(5)) +
                ' is added at ' +
                cast(Getdate() as nvarchar(20))
              )
END
```

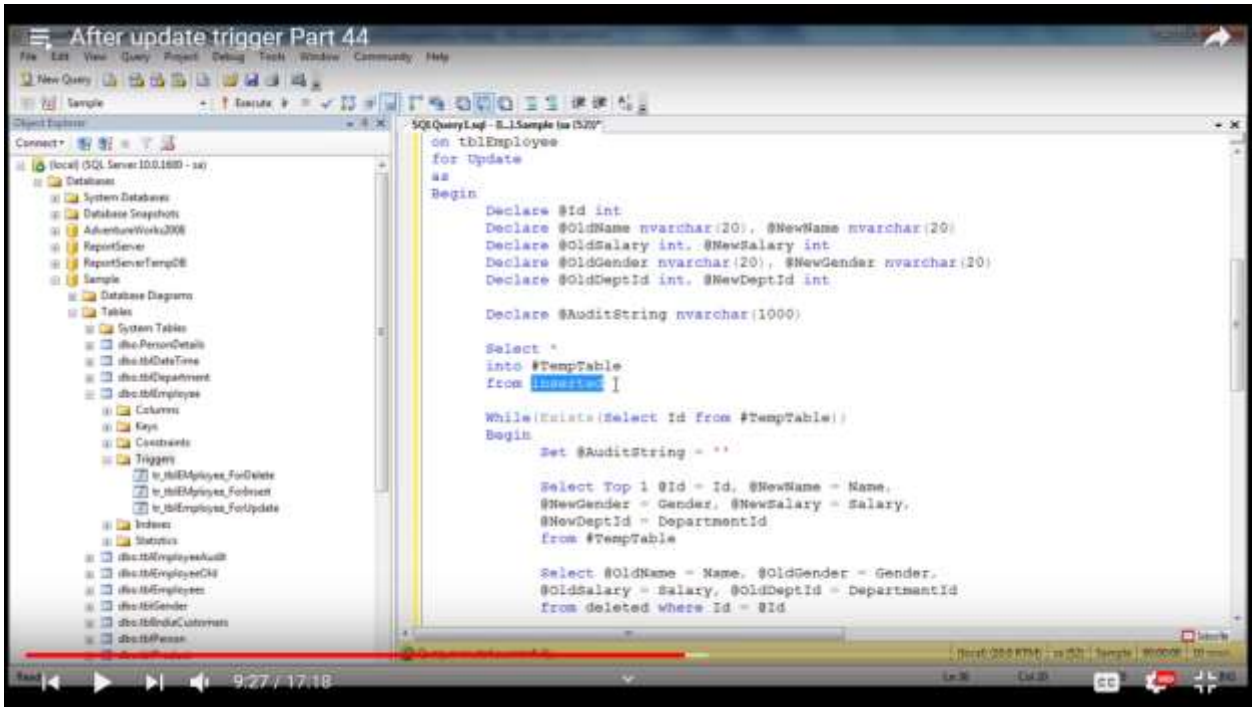
```
CREATE TRIGGER tr_tblEmployee_ForDelete
ON tblEmployee
FOR DELETE
AS
BEGIN
        Declare @Id int
        Select @Id = Id from deleted

        insert into tblEmployeeAudit
        values('An existing employee with Id = ' +
                Cast(@Id as nvarchar(5)) +
                ' is deleted at ' +
                cast(Getdate() as nvarchar(20))
              )
END
```

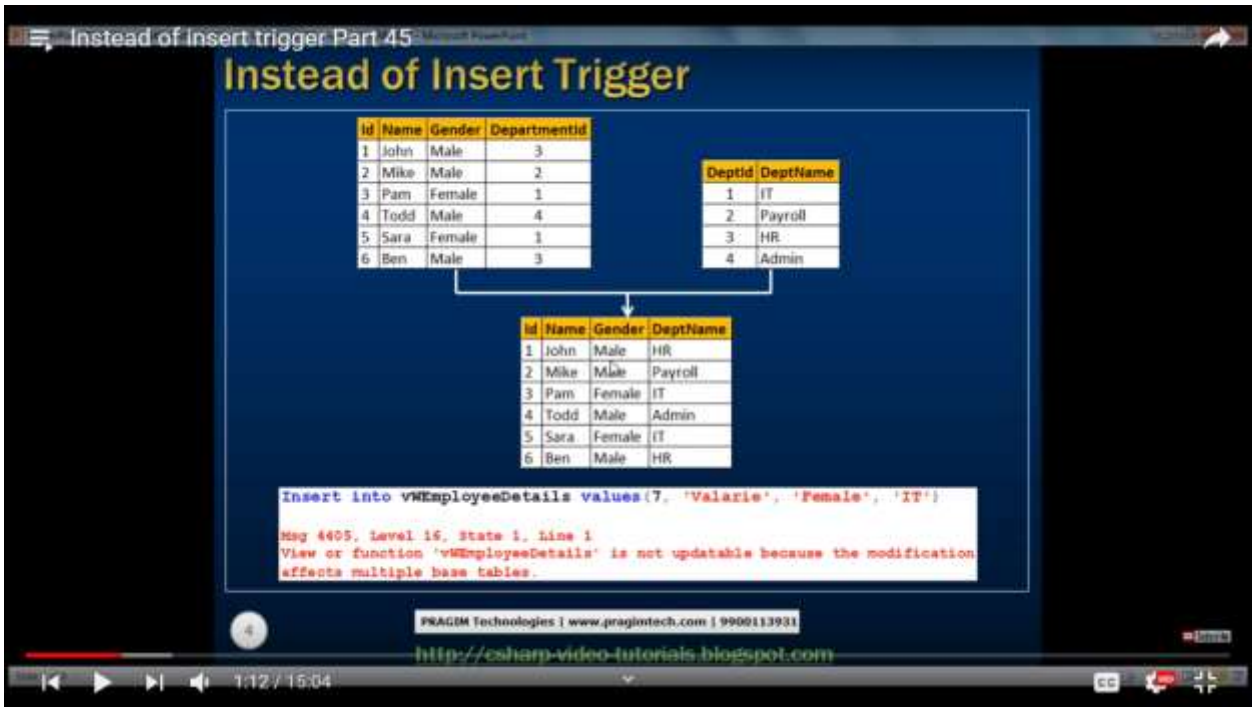
```
Insert into tblEmployee values
(8,'Ben', 4800, 'Male', 3)
```

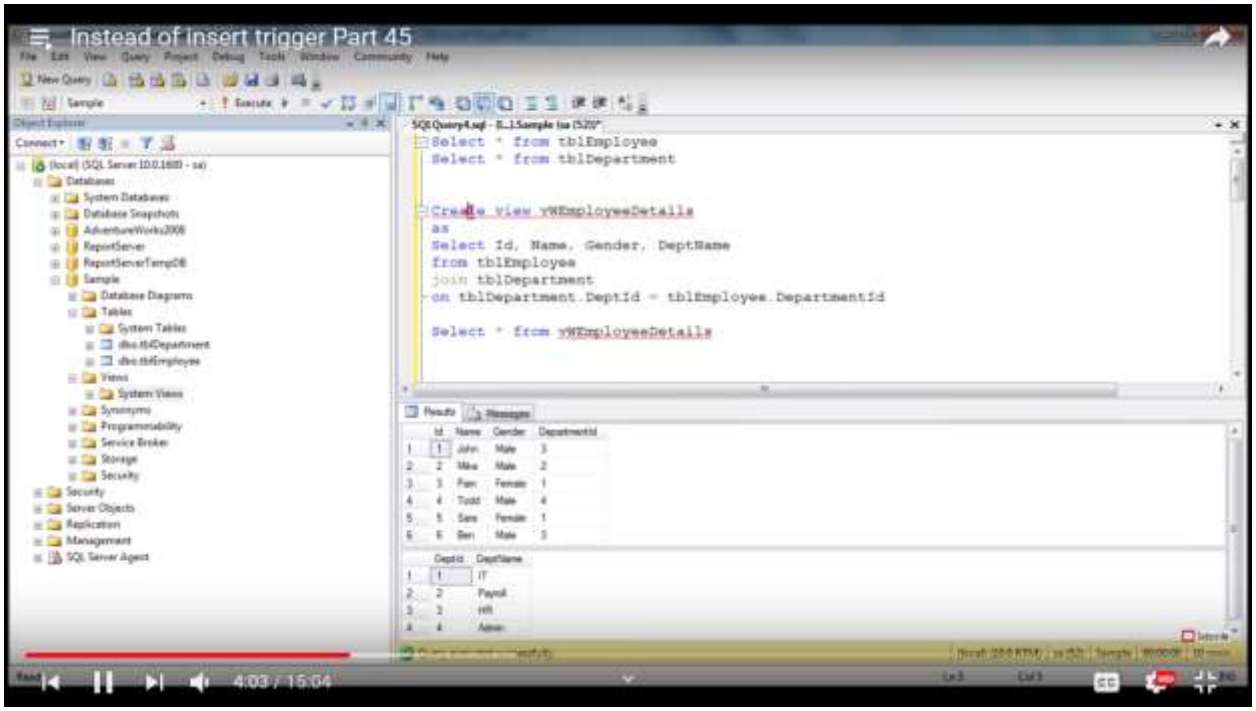
```
Delete from tblEmployee where Id = 3
```

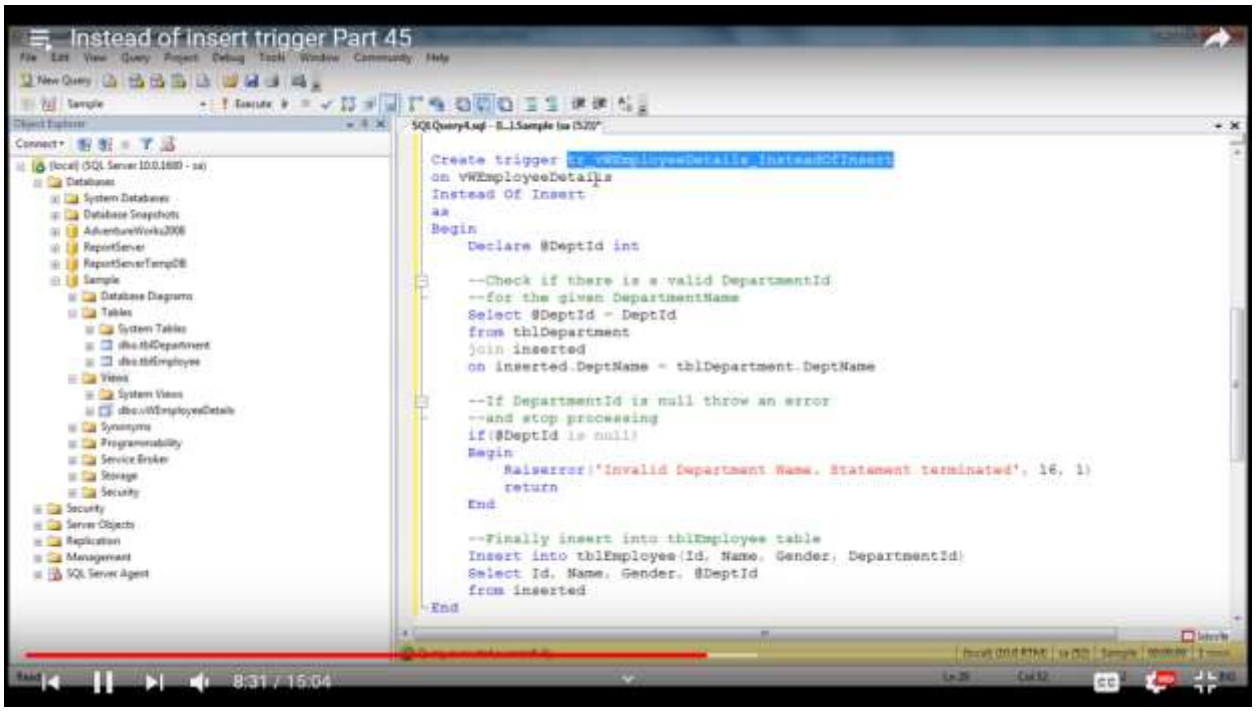

DML triggers in sql server Part 43
After Triggers
Id Name Salary Gender DepartmentId
1 John 5000 Male 3
2 Mike 3400 Male 2
3 Pam 6000 Female 1
4 Todd 4800 Male 4
5 Sara 3200 Female 1
6 Ben 4800 Male 3
Id AuditData
1 New employee with Id = 8 is added at Sep 17 2012 8:04PM
2 An existing employee with Id = 8 is deleted at Sep 17 2012 8:29PM
3 An existing employee with Id = 1 is deleted at Sep 17 2012 8:31PM
4 An existing employee with Id = 3 is deleted at Sep 17 2012 8:31PM
5 An existing employee with Id = 4 is deleted at Sep 17 2012 8:31PM
CREATE TRIGGER tr_tblEmployee_ForInsert
ON tblEmployee
FOR INSERT
AS
BEGIN
Declare @Id int
Select @Id = Id from inserted
insert into tblEmployeeAudit
values ('New employee with Id = ' +
Cast(@Id as nvarchar(5)) +
' is added at ' +
cast(Getdate() as nvarchar(20))
)
END
CREATE TRIGGER tr_tblEmployee_ForDelete
ON tblEmployee
FOR DELETE
AS
BEGIN
Declare @Id int
Select @Id = Id from deleted
insert into tblEmployeeAudit
values('An existing employee with Id = ' +
Cast(@Id as nvarchar(5)) +
' is deleted at ' +
cast(Getdate() as nvarchar(20))
)
END
Insert into tblEmployee values
(8,'Ben', 4800, 'Male', 3)
Delete from tblEmployee where Id = 3
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
3:14 / 17:44

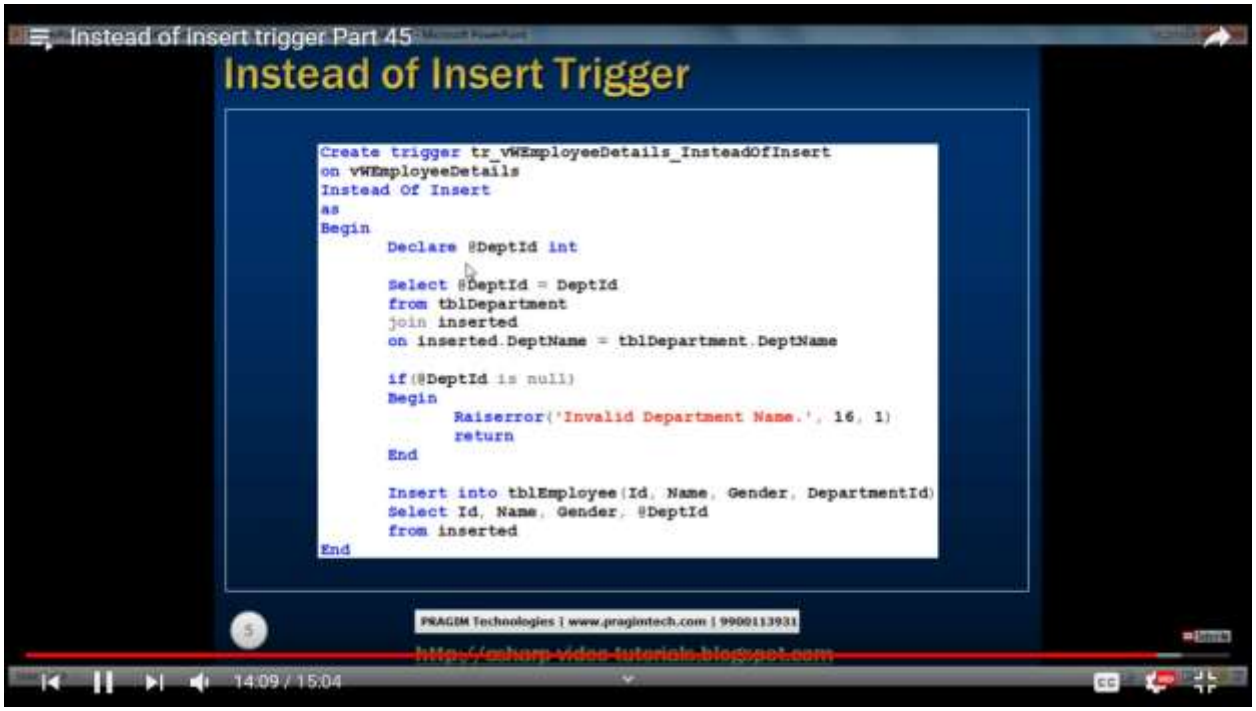

After update trigger Part 44
After Update Trigger
Id Name Salary Gender DepartmentId
1 John 5000 Male 3
2 Mike 3400 Male 2
3 Pam 6000 Female 1
4 Todd 4800 Male 4
5 Sara 3200 Female 1
6 Ben 4800 Male 3
Id AuditData
1 New employee with Id = 9 is added at Sep 17 2012 9:54PM
2 An existing employee with Id = 1 is deleted at Sep 17 2012 9:57PM
3 Employee with Id = 2 changed NAME from Mike to Mikey SALARY from 3400 to 3500
Note: The After trigger for UPDATE event, makes use of both inserted and deleted tables. The inserted table contains the updated data and the deleted table contains the old data.
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
1:55 / 17:18

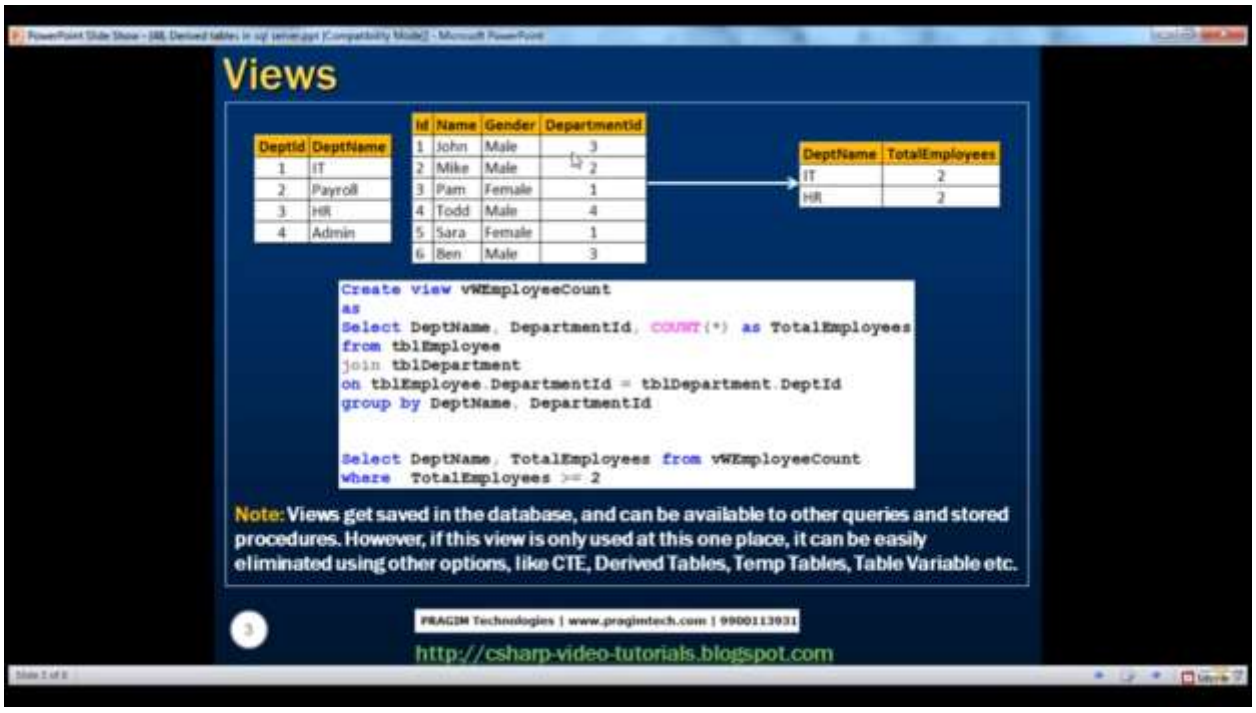

After update trigger Part 44
on tblEmployee
for Update
as
Begin
Declare @Id int
Declare @OldName nvarchar(20), @NewName nvarchar(20)
Declare @OldSalary int, @NewSalary int
Declare @OldGender nvarchar(20), @NewGender nvarchar(20)
Declare @OldDeptId int, @NewDeptId int
Declare @AuditString nvarchar(1000)
Select *
into #TempTable
from inserted
While(Exists(Select Id from #TempTable))
Begin
Set @AuditString = ''
Select Top 1 @Id = Id, @NewName = Name,
@NewGender = Gender, @NewSalary = Salary,
@NewDeptId = DepartmentId
from #TempTable
Select @OldName = Name, @OldGender = Gender,
@OldSalary = Salary, @OldDeptId = DepartmentId
from deleted where Id = @Id
9:27 / 17:18


Instead of insert trigger Part 45
Instead of Insert Trigger
Id Name Gender DepartmentId
1 John Male 3
2 Mike Male 2
3 Pam Female 1
4 Todd Male 4
5 Sara Female 1
6 Ben Male 3
DeptId DeptName
1 IT
2 Payroll
3 HR
4 Admin
Id Name Gender DeptName
1 John Male HR
2 Mike Male Payroll
3 Pam Female IT
4 Todd Male Admin
5 Sara Female IT
6 Ben Male HR
Insert into vWEmployeeDetails values(7, 'Valarie', 'Female', 'IT')
Msg 4405, Level 16, State 1, Line 1
View or function 'vWEmployeeDetails' is not updatable because the modification
affects multiple base tables.
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
1:12 / 15:04


Instead of insert trigger Part 45
Select * from tblEmployee
Select * from tblDepartment
Create view vWEmployeeDetails
as
Select Id, Name, Gender, DeptName
from tblEmployee
join tblDepartment
on tblDepartment.DeptId = tblEmployee.DepartmentId
Select * from vWEmployeeDetails
4:03 / 15:04


Instead of insert trigger Part 45
Create trigger tr_vWEmployeeDetails_InsteadOfInsert
on vWEmployeeDetails
Instead Of Insert
as
Begin
Declare @DeptId int
--Check if there is a valid DepartmentId
--for the given DepartmentName
Select @DeptId = DeptId
from tblDepartment
join inserted
on inserted.DeptName = tblDepartment.DeptName
--If DepartmentId is null throw an error
--and stop processing
if(@DeptId is null)
Begin
Raiserror('Invalid Department Name. Statement terminated', 16, 1)
return
End
--Finally insert into tblEmployee table
Insert into tblEmployee(Id, Name, Gender, DepartmentId)
Select Id, Name, Gender, @DeptId
from inserted
End
8:31 / 15:04


Instead of insert trigger Part 45
Instead of Insert Trigger
Create trigger tr_vWEmployeeDetails_InsteadOfInsert
on vWEmployeeDetails
Instead Of Insert
as
Begin
Declare @DeptId int
Select @DeptId = DeptId
from tblDepartment
join inserted
on inserted.DeptName = tblDepartment.DeptName
if(@DeptId is null)
Begin
Raiserror('Invalid Department Name.', 16, 1)
return
End
Insert into tblEmployee(Id, Name, Gender, DepartmentId)
Select Id, Name, Gender, @DeptId
from inserted
End
5
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
14:09 / 15:04


Views
DeptId DeptName
1 IT
2 Payroll
3 HR
4 Admin
Id Name Gender DepartmentId
1 John Male 3
2 Mike Male 2
3 Pam Female 1
4 Todd Male 4
5 Sara Female 1
6 Ben Male 3
DeptName TotalEmployees
IT 2
HR 2
Create view vWEmployeeCount
as
Select DeptName, DepartmentId, COUNT(*) as TotalEmployees
from tblEmployee
join tblDepartment
on tblEmployee.DepartmentId = tblDepartment.DeptId
group by DeptName, DepartmentId
Select DeptName, TotalEmployees from vWEmployeeCount
where TotalEmployees >= 2
Note: Views get saved in the database, and can be available to other queries and stored procedures. However, if this view is only used at this one place, it can be easily eliminated using other options, like CTE, Derived Tables, Temp Tables, Table Variable etc.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com

Temp Tables

DeptId	DeptName
1	IT
2	Payroll
3	HR
4	Admin

Id	Name	Gender	DepartmentId
1	John	Male	3
2	Mike	Male	2
3	Pam	Female	1
4	Todd	Male	4
5	Sara	Female	1
6	Ben	Male	3

DeptName	TotalEmployees
IT	2
HR	2

```
Select DeptName, DepartmentId, COUNT(*) as TotalEmployees
into #TempEmployeeCount
from tblEmployee
join tblDepartment
on tblEmployee.DepartmentId = tblDepartment.DeptId
group by DeptName, DepartmentId

Select DeptName, TotalEmployees
From #TempEmployeeCount
where  TotalEmployees >= 2

Drop Table #TempEmployeeCount
```

Notes: Temporary tables are stored in TempDB. Local temporary tables are visible only in the current session, and can be shared between nested stored procedure calls. Global temporary tables are visible to other sessions and are destroyed, when the last connection referencing the table is closed.

Table Variable

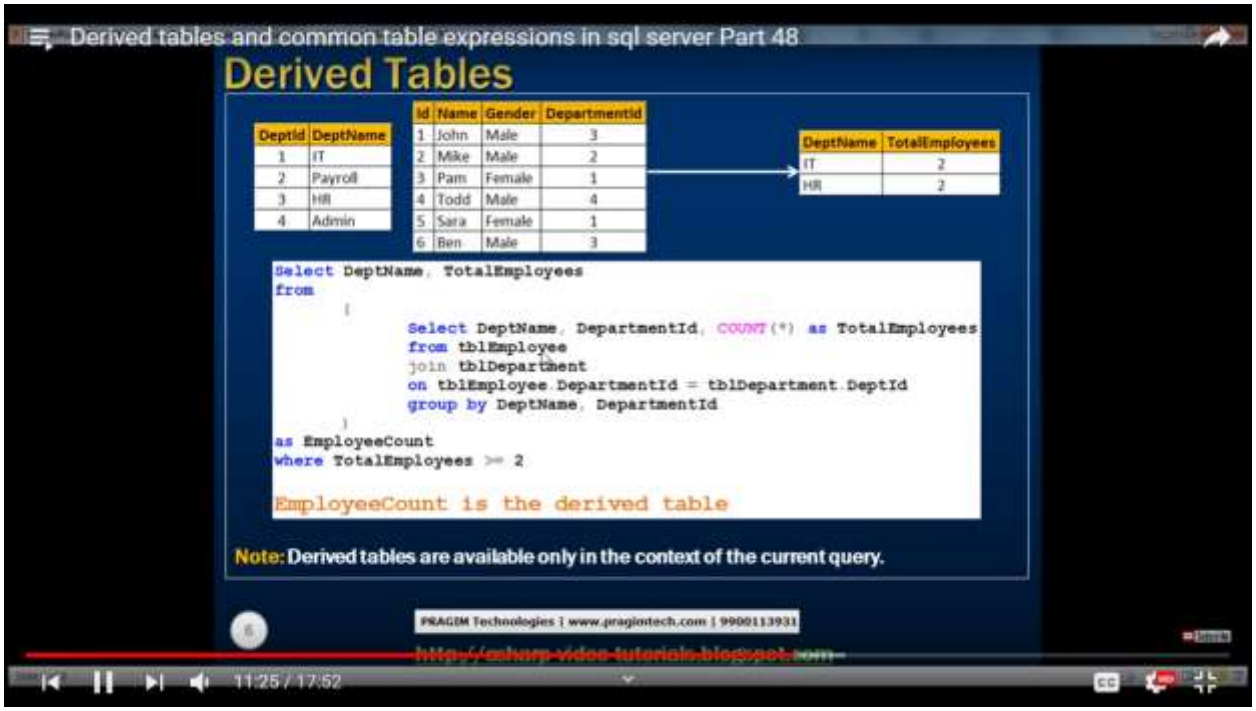
DeptId	DeptName
1	IT
2	Payroll
3	HR
4	Admin

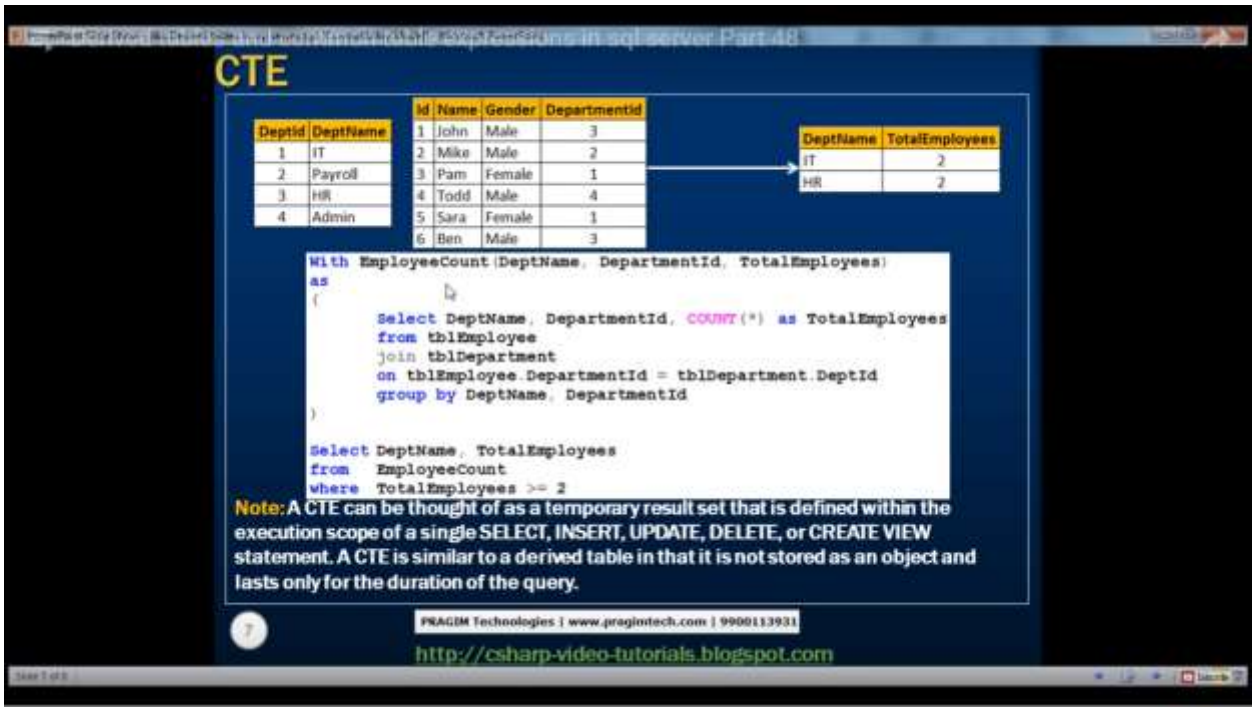
Id	Name	Gender	DepartmentId
1	John	Male	3
2	Mike	Male	2
3	Pam	Female	1
4	Todd	Male	4
5	Sara	Female	1
6	Ben	Male	3

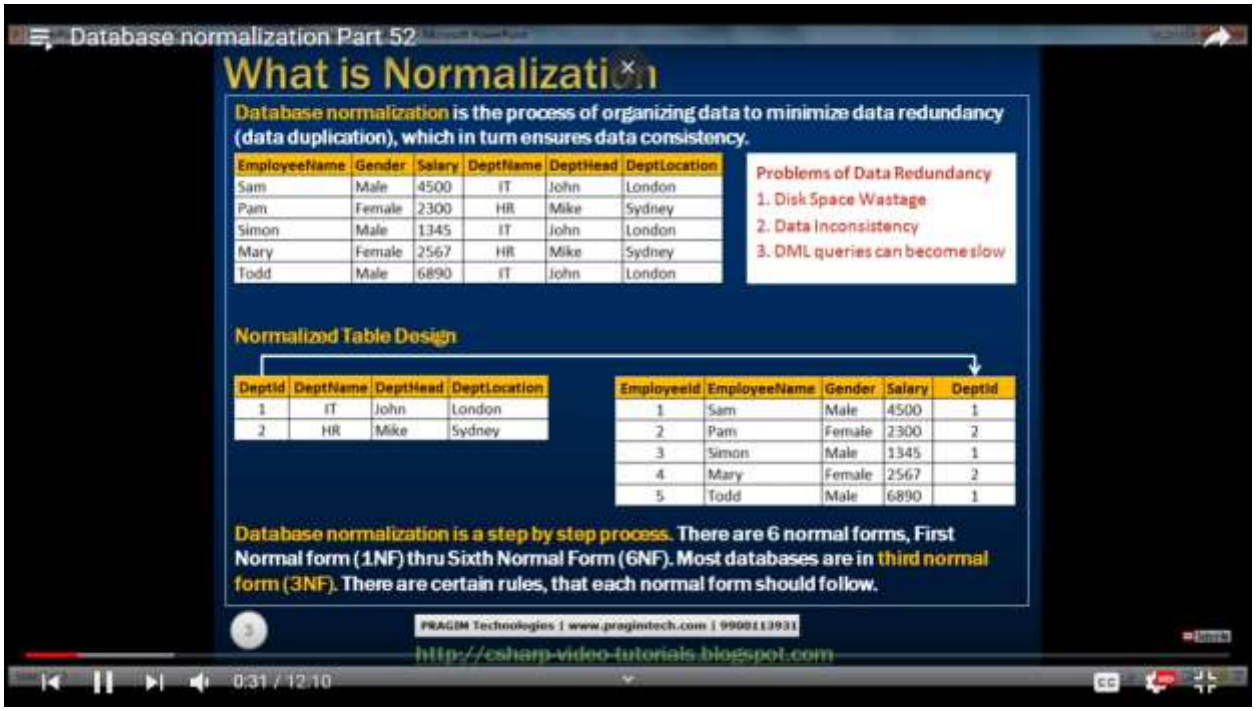
DeptName	TotalEmployees
IT	2
HR	2

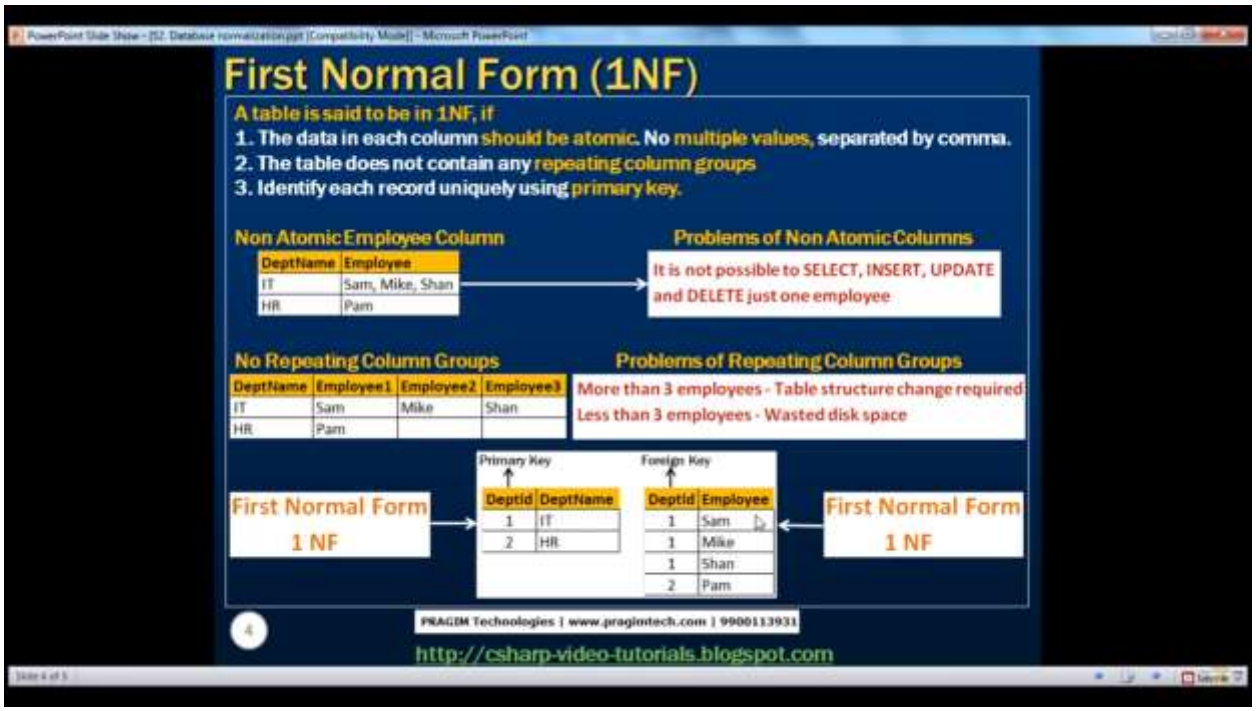
```
Declare @tblEmployeeCount table(DeptName nvarchar(20),DepartmentId int,
TotalEmployees int)

Insert @tblEmployeeCount
Select DeptName, DepartmentId, COUNT(*) as TotalEmployees
from tblEmployee
join tblDepartment
on tblEmployee.DepartmentId = tblDepartment.DeptId
group by DeptName, DepartmentId

Select DeptName, TotalEmployees
From @tblEmployeeCount
where  TotalEmployees >= 2
```

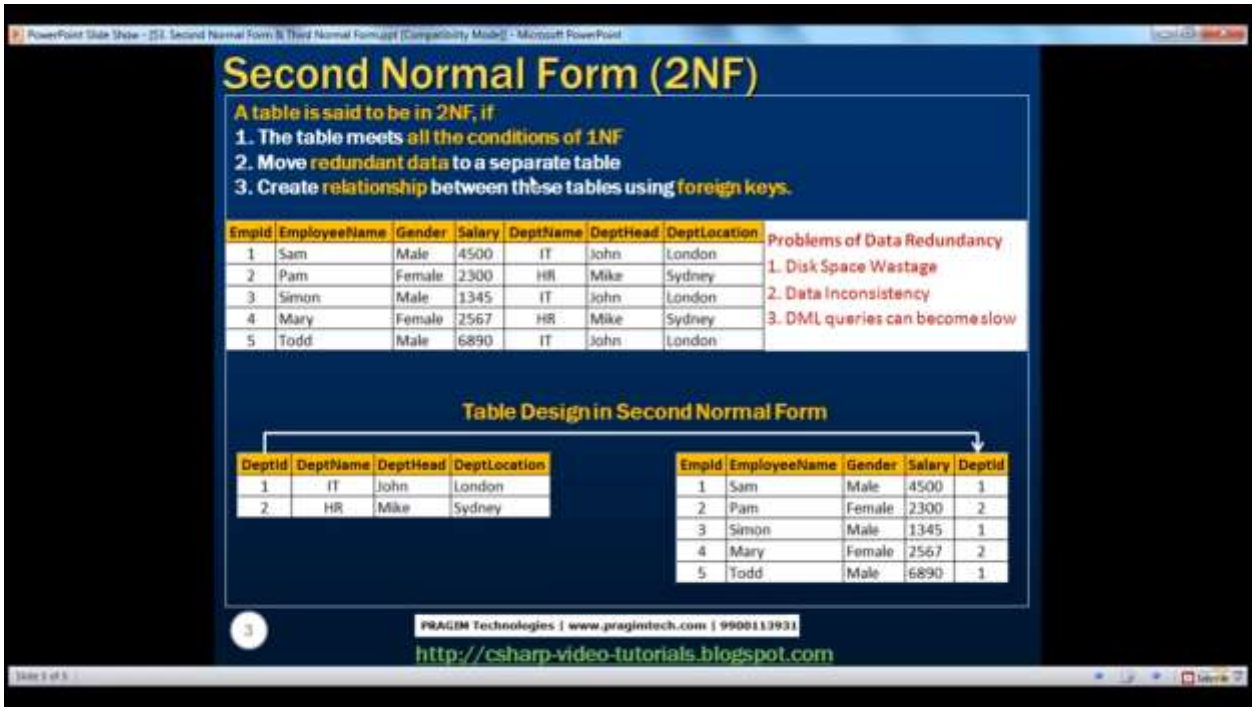
Note: Just like TempTables, a table variable is also created in TempDB. The scope of a table variable is the batch, stored procedure, or statement block in which it is declared. They can be passed as parameters between procedures.


Derived tables and common table expressions in sql server Part 48
Derived Tables
DeptId DeptName
1 IT
2 Payroll
3 HR
4 Admin
Id Name Gender DepartmentId
1 John Male 3
2 Mike Male 2
3 Pam Female 1
4 Todd Male 4
5 Sara Female 1
6 Ben Male 3
DeptName TotalEmployees
IT 2
HR 2
Select DeptName, TotalEmployees
from
(
Select DeptName, DepartmentId, COUNT(*) as TotalEmployees
from tblEmployee
join tblDepartment
on tblEmployee.DepartmentId = tblDepartment.DeptId
group by DeptName, DepartmentId
)
as EmployeeCount
where TotalEmployees >= 2
EmployeeCount is the derived table
Note: Derived tables are available only in the context of the current query.
6
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
11:25 / 17:52


CTE
DeptId DeptName
1 IT
2 Payroll
3 HR
4 Admin
Id Name Gender DepartmentId
1 John Male 3
2 Mike Male 2
3 Pam Female 1
4 Todd Male 4
5 Sara Female 1
6 Ben Male 3
DeptName TotalEmployees
IT 2
HR 2
With EmployeeCount(DeptName, DepartmentId, TotalEmployees)
as
(
Select DeptName, DepartmentId, COUNT(*) as TotalEmployees
from tblEmployee
join tblDepartment
on tblEmployee.DepartmentId = tblDepartment.DeptId
group by DeptName, DepartmentId
)
Select DeptName, TotalEmployees
from EmployeeCount
where TotalEmployees >= 2
Note: A CTE can be thought of as a temporary result set that is defined within the execution scope of a single SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. A CTE is similar to a derived table in that it is not stored as an object and lasts only for the duration of the query.
7
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com


Database normalization Part 52
What is Normalization
Database normalization is the process of organizing data to minimize data redundancy (data duplication), which in turn ensures data consistency.
EmployeeName | Gender | Salary | DeptName | DeptHead | DeptLocation
Sam | Male | 4500 | IT | John | London
Pam | Female | 2300 | HR | Mike | Sydney
Simon | Male | 1345 | IT | John | London
Mary | Female | 2567 | HR | Mike | Sydney
Todd | Male | 6890 | IT | John | London
Problems of Data Redundancy
1. Disk Space Wastage
2. Data Inconsistency
3. DML queries can become slow
Normalized Table Design
DeptId | DeptName | DeptHead | DeptLocation
1 | IT | John | London
2 | HR | Mike | Sydney
EmployeeId | EmployeeName | Gender | Salary | DeptId
1 | Sam | Male | 4500 | 1
2 | Pam | Female | 2300 | 2
3 | Simon | Male | 1345 | 1
4 | Mary | Female | 2567 | 2
5 | Todd | Male | 6890 | 1
Database normalization is a step by step process. There are 6 normal forms, First Normal form (1NF) thru Sixth Normal Form (6NF). Most databases are in third normal form (3NF). There are certain rules, that each normal form should follow.
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:31 / 12:10

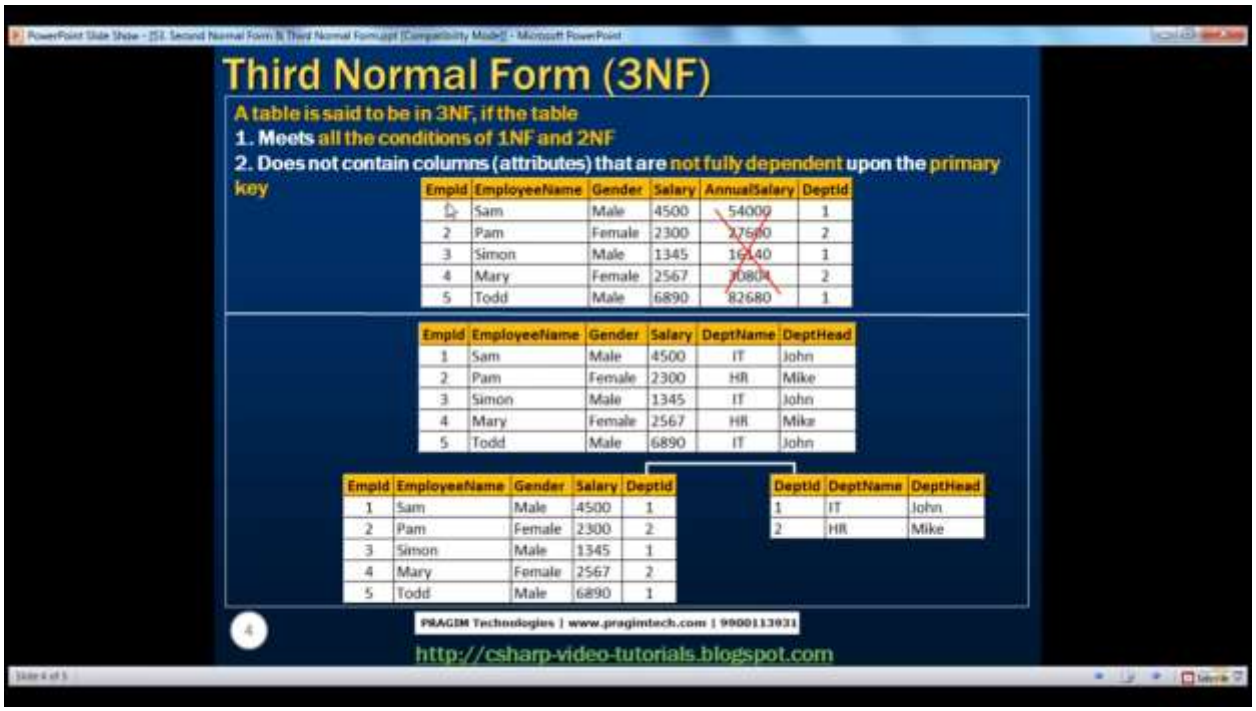

First Normal Form (1NF)
A table is said to be in 1NF, if
1. The data in each column should be atomic. No multiple values, separated by comma.
2. The table does not contain any repeating column groups
3. Identify each record uniquely using primary key.
Non Atomic Employee Column
DeptName | Employee
IT | Sam, Mike, Shan
HR | Pam
Problems of Non Atomic Columns
It is not possible to SELECT, INSERT, UPDATE and DELETE just one employee
No Repeating Column Groups
DeptName | Employee1 | Employee2 | Employee3
IT | Sam | Mike | Shan
HR | Pam
Problems of Repeating Column Groups
More than 3 employees - Table structure change required
Less than 3 employees - Wasted disk space
First Normal Form 1 NF
Primary Key
Deptid | DeptName
1 | IT
2 | HR
Foreign Key
Deptid | Employee
1 | Sam
1 | Mike
1 | Shan
2 | Pam
First Normal Form 1 NF
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com



EmpId	EmployeeName	Gender	Salary	DeptName	DeptHead	DeptLocation
1	Sam	Male	4500	IT	John	London
2	Pam	Female	2300	HR	Mike	Sydney
3	Simon	Male	1345	IT	John	London
4	Mary	Female	2567	HR	Mike	Sydney
5	Todd	Male	6890	IT	John	London

DeptId	DeptName	DeptHead	DeptLocation
1	IT	John	London
2	HR	Mike	Sydney

EmpId	EmployeeName	Gender	Salary	DeptId
1	Sam	Male	4500	1
2	Pam	Female	2300	2
3	Simon	Male	1345	1
4	Mary	Female	2567	2
5	Todd	Male	6890	1

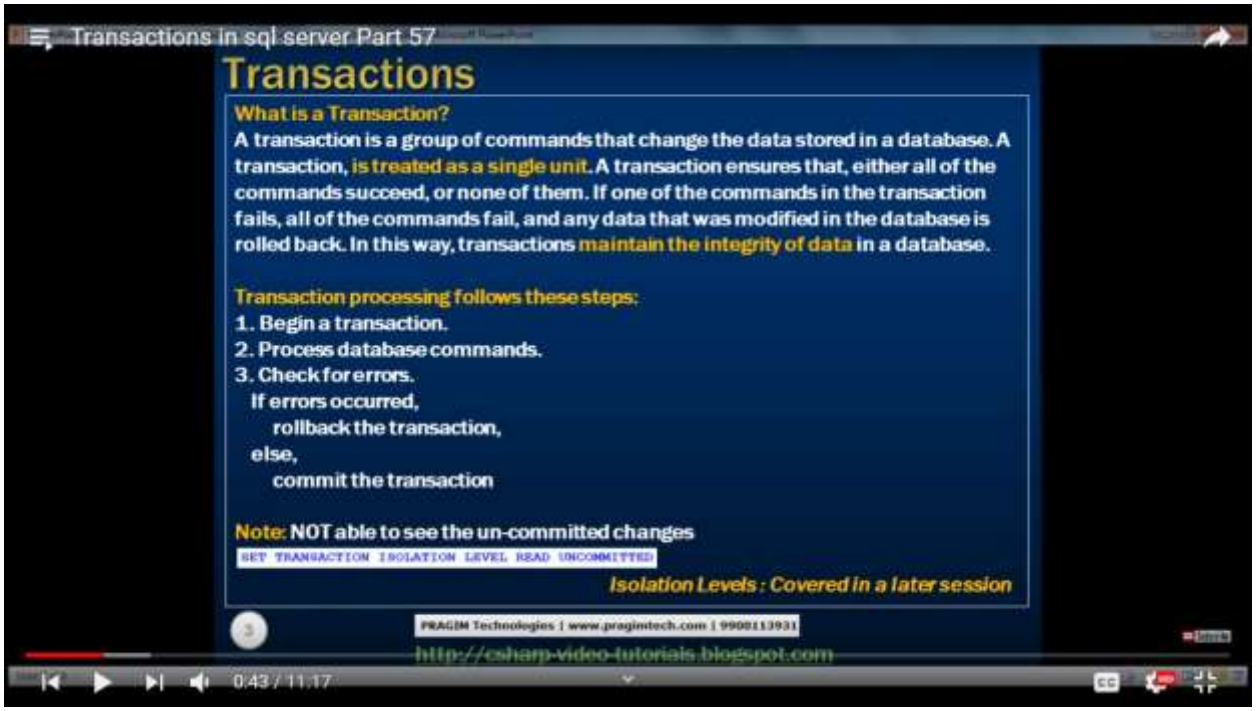


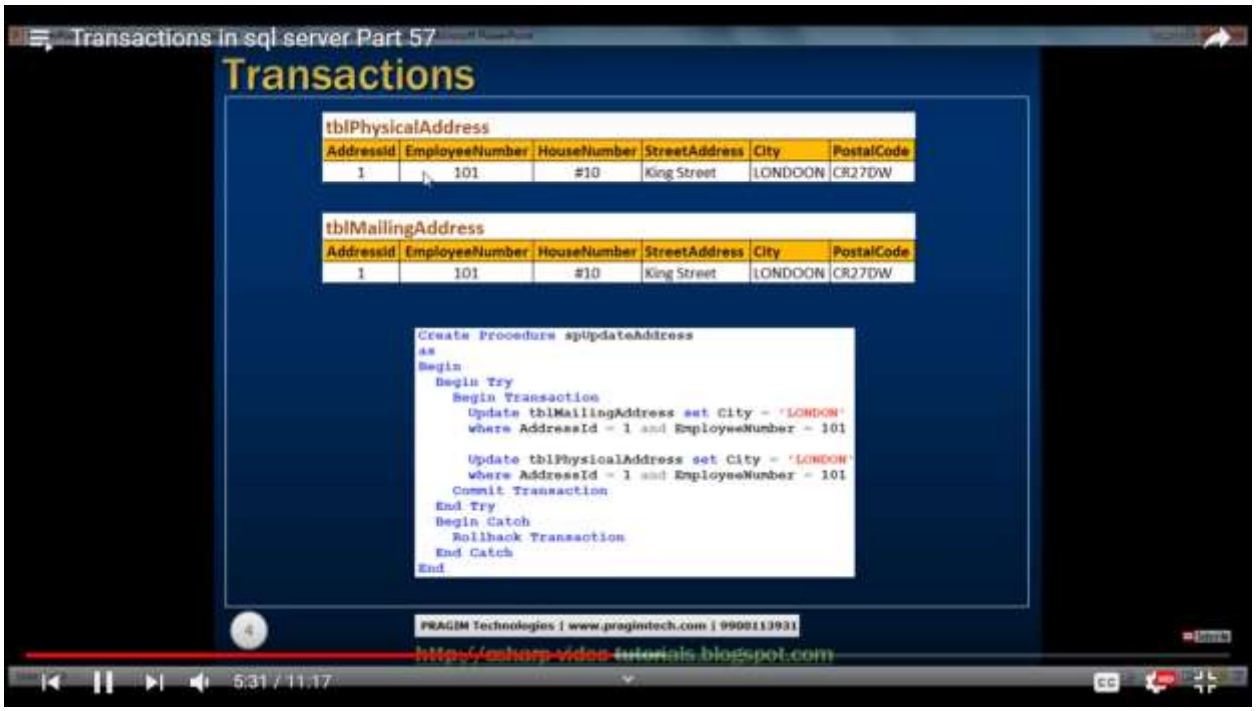
EmpId	EmployeeName	Gender	Salary	AnnualSalary	DeptId
1	Sam	Male	4500	54000	1
2	Pam	Female	2300	27600	2
3	Simon	Male	1345	16140	1
4	Mary	Female	2567	30804	2
5	Todd	Male	6890	82680	1

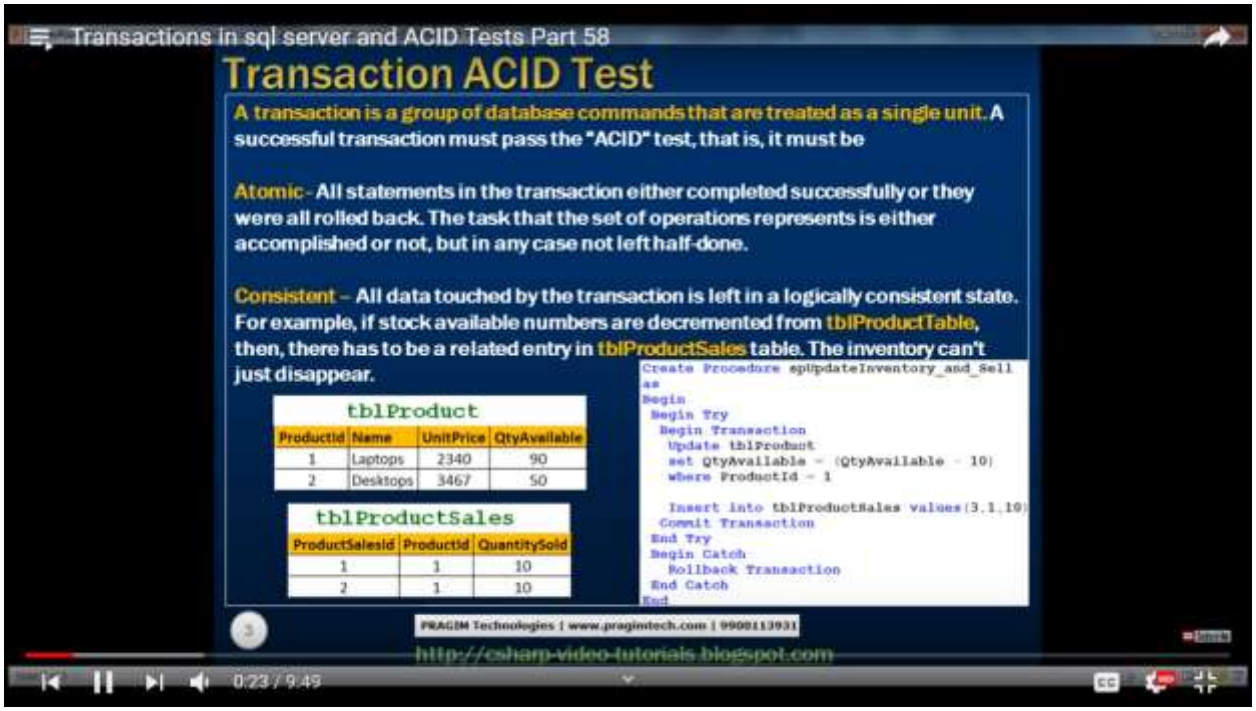
EmpId	EmployeeName	Gender	Salary	DeptName	DeptHead
1	Sam	Male	4500	IT	John
2	Pam	Female	2300	HR	Mike
3	Simon	Male	1345	IT	John
4	Mary	Female	2567	HR	Mike
5	Todd	Male	6890	IT	John

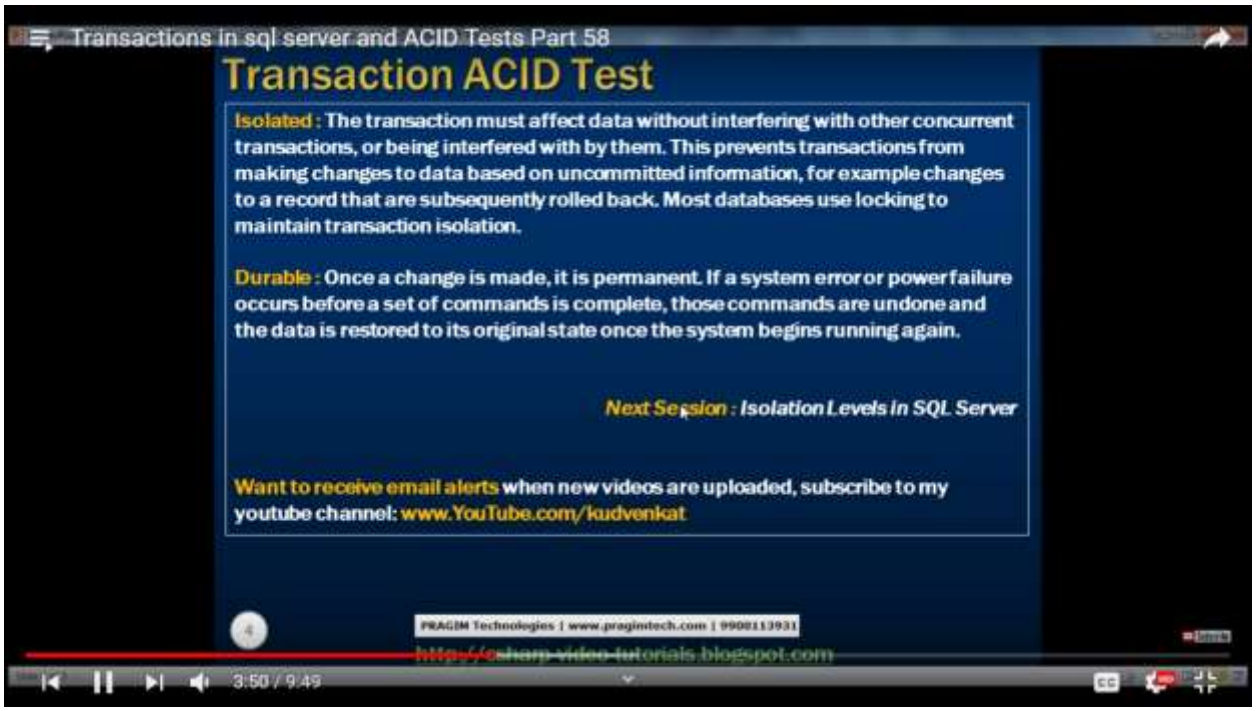
EmpId	EmployeeName	Gender	Salary	DeptId
1	Sam	Male	4500	1
2	Pam	Female	2300	2
3	Simon	Male	1345	1
4	Mary	Female	2567	2
5	Todd	Male	6890	1

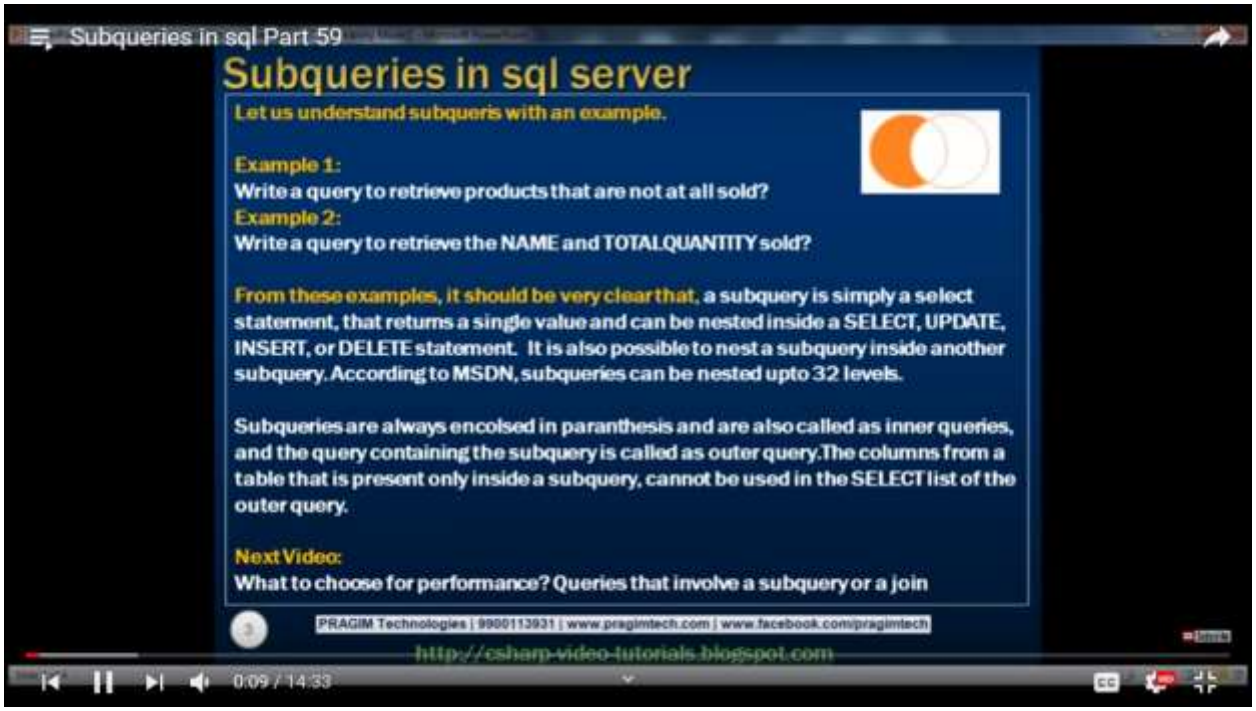
DeptId	DeptName	DeptHead
1	IT	John
2	HR	Mike

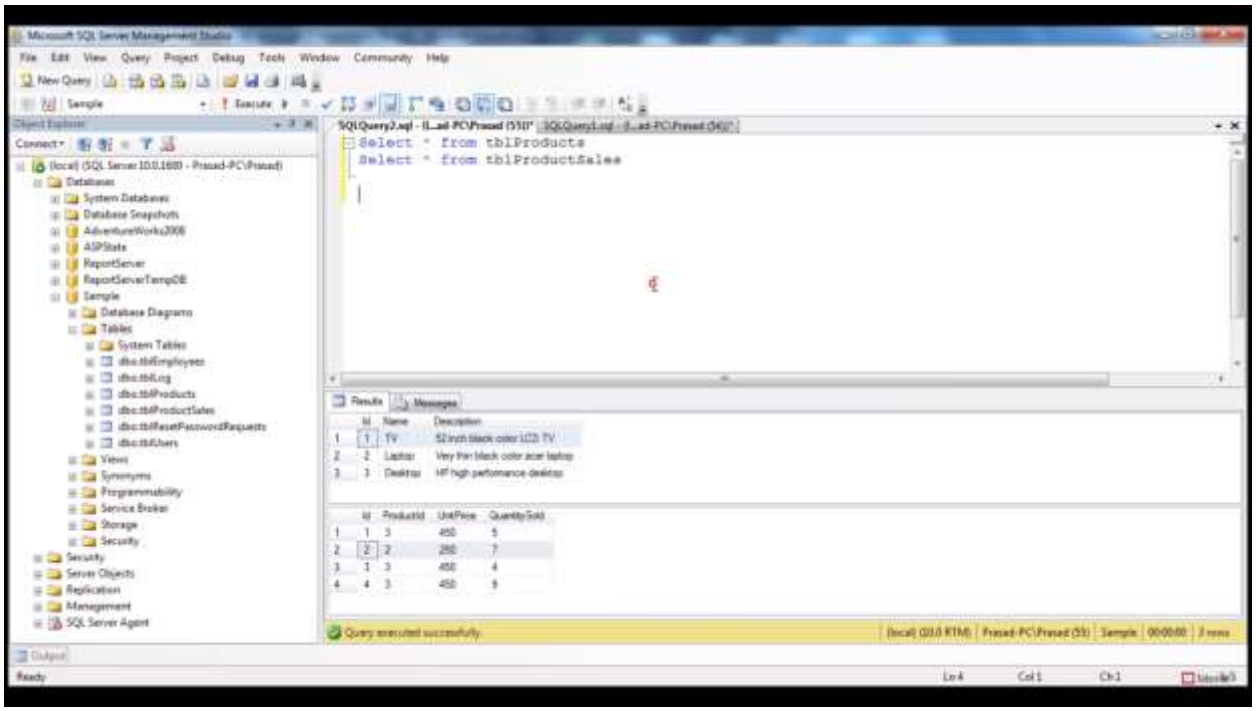

Transactions in sql server Part 57
Transactions
What is a Transaction?
A transaction is a group of commands that change the data stored in a database. A transaction, is treated as a single unit. A transaction ensures that, either all of the commands succeed, or none of them. If one of the commands in the transaction fails, all of the commands fail, and any data that was modified in the database is rolled back. In this way, transactions maintain the integrity of data in a database.
Transaction processing follows these steps:
1. Begin a transaction.
2. Process database commands.
3. Check for errors.
If errors occurred,
rollback the transaction,
else,
commit the transaction
Note: NOT able to see the un-committed changes
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
Isolation Levels : Covered in a later session
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:43 / 11:17

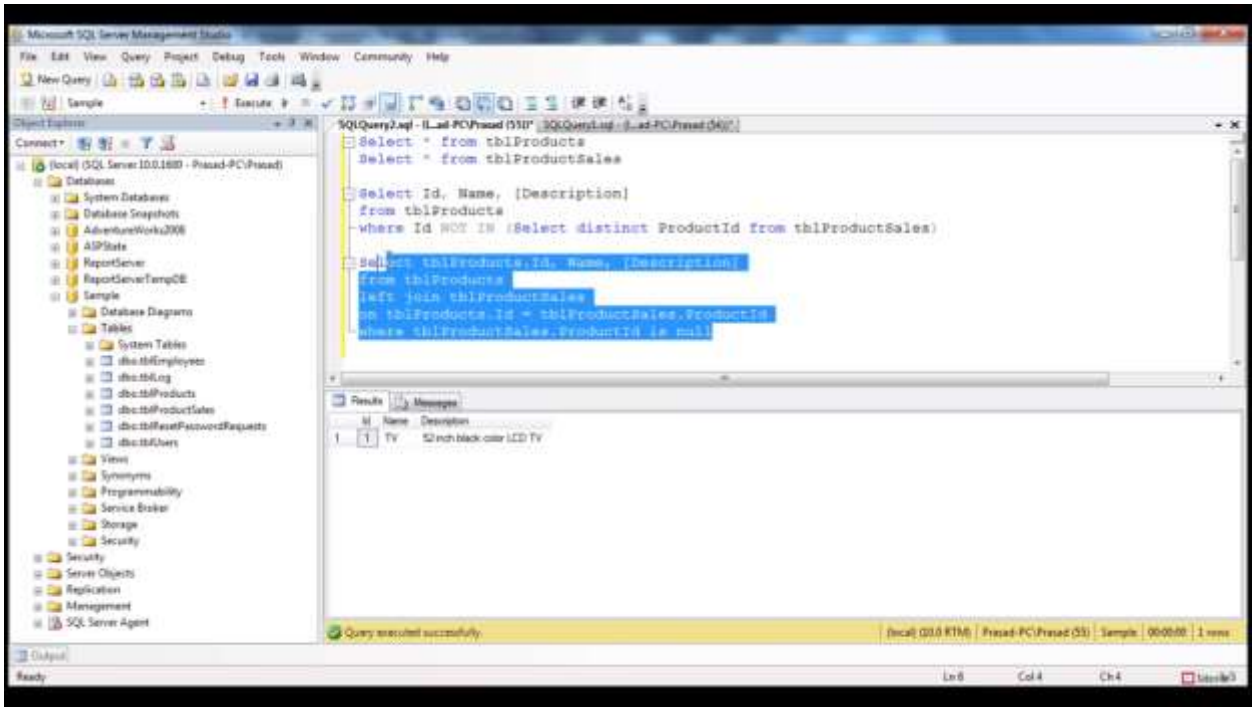

Transactions in sql server Part 57
Transactions
tblPhysicalAddress
AddressId | EmployeeNumber | HouseNumber | StreetAddress | City | PostalCode
1 | 101 | #10 | King Street | LONDOON | CR27DW
tblMailingAddress
AddressId | EmployeeNumber | HouseNumber | StreetAddress | City | PostalCode
1 | 101 | #10 | King Street | LONDOON | CR27DW
Create Procedure spUpdateAddress
as
Begin
Begin Try
Begin Transaction
Update tblMailingAddress set City = 'LONDON'
where AddressId = 1 and EmployeeNumber = 101
Update tblPhysicalAddress set City = 'LONDON'
where AddressId = 1 and EmployeeNumber = 101
Commit Transaction
End Try
Begin Catch
Rollback Transaction
End Catch
End
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
5:31 / 11:17

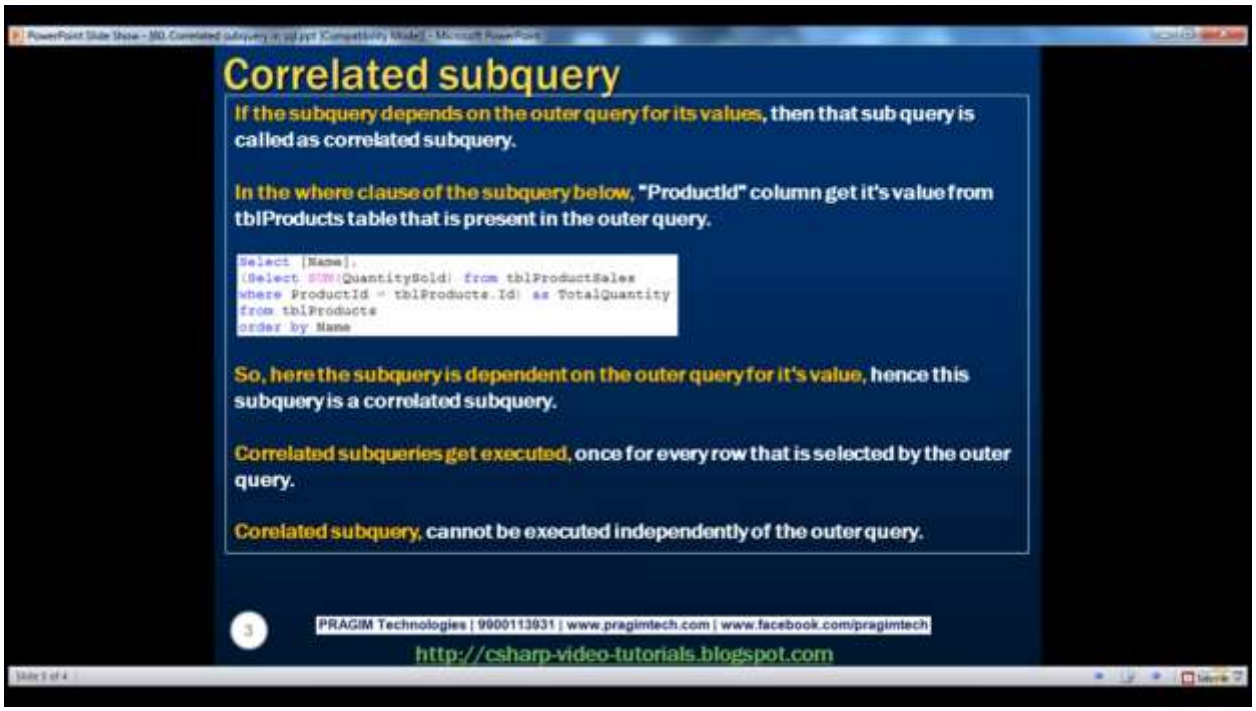

Transactions in sql server and ACID Tests Part 58
Transaction ACID Test
A transaction is a group of database commands that are treated as a single unit. A successful transaction must pass the "ACID" test, that is, it must be
Atomic - All statements in the transaction either completed successfully or they were all rolled back. The task that the set of operations represents is either accomplished or not, but in any case not left half-done.
Consistent – All data touched by the transaction is left in a logically consistent state. For example, if stock available numbers are decremented from tblProductTable, then, there has to be a related entry in tblProductSales table. The inventory can't just disappear.
tblProduct
ProductId Name UnitPrice QtyAvailable
1 Laptops 2340 90
2 Desktops 3467 50
tblProductSales
ProductSalesId ProductId QuantitySold
1 1 10
2 1 10
Create Procedure spUpdateInventory_and_Sell
as
Begin
Begin Try
Begin Transaction
Update tblProduct
set QtyAvailable = (QtyAvailable - 10)
where ProductId = 1
Insert into tblProductSales values(3,1,10)
Commit Transaction
End Try
Begin Catch
Rollback Transaction
End Catch
End
3
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
0:23 / 9:49


Transactions in sql server and ACID Tests Part 58
Transaction ACID Test
Isolated : The transaction must affect data without interfering with other concurrent transactions, or being interfered with by them. This prevents transactions from making changes to data based on uncommitted information, for example changes to a record that are subsequently rolled back. Most databases use locking to maintain transaction isolation.
Durable : Once a change is made, it is permanent. If a system error or power failure occurs before a set of commands is complete, those commands are undone and the data is restored to its original state once the system begins running again.
Next Session : Isolation Levels in SQL Server
Want to receive email alerts when new videos are uploaded, subscribe to my youtube channel: www.YouTube.com/kudvenkat
4
PRAGIM Technologies | www.pragimtech.com | 9900113931
http://csharp-video-tutorials.blogspot.com
3:50 / 9:49

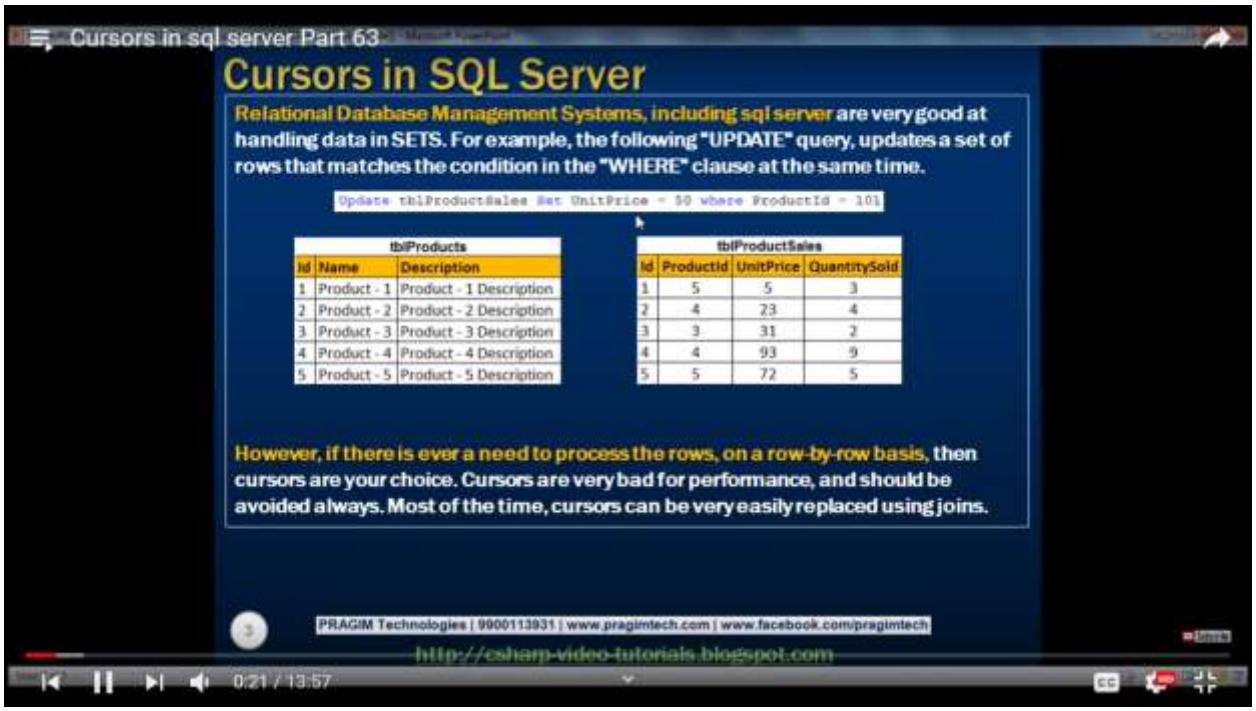

Subqueries in sql Part 59
Subqueries in sql server
Let us understand subqueris with an example.
Example 1:
Write a query to retrieve products that are not at all sold?
Example 2:
Write a query to retrieve the NAME and TOTALQUANTITY sold?
From these examples, it should be very clear that, a subquery is simply a select statement, that returns a single value and can be nested inside a SELECT, UPDATE, INSERT, or DELETE statement. It is also possible to nest a subquery inside another subquery. According to MSDN, subqueries can be nested upto 32 levels.
Subqueries are always encolsed in paranthesis and are also called as inner queries, and the query containing the subquery is called as outer query.The columns from a table that is present only inside a subquery, cannot be used in the SELECT list of the outer query.
Next Video:
What to choose for performance? Queries that involve a subquery or a join
PRAGIM Technologies | 9900113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com
0:09 / 14:33

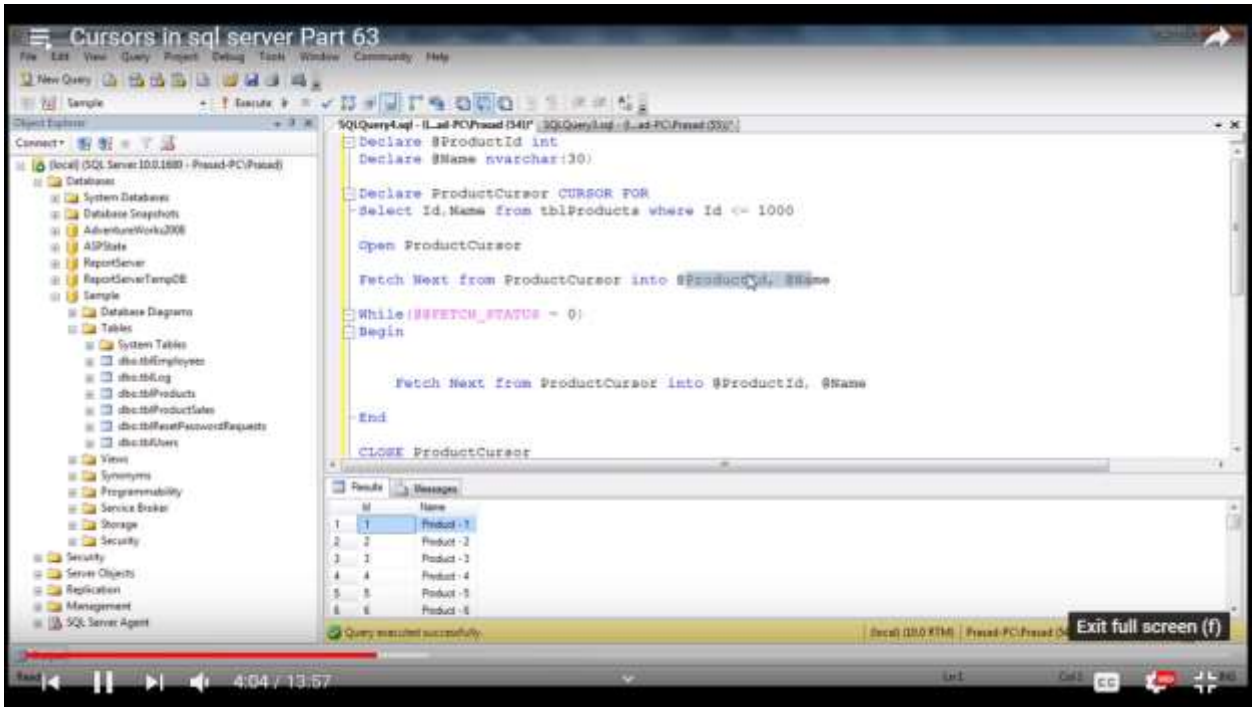

Select * from tblProducts
Select * from tblProductSales

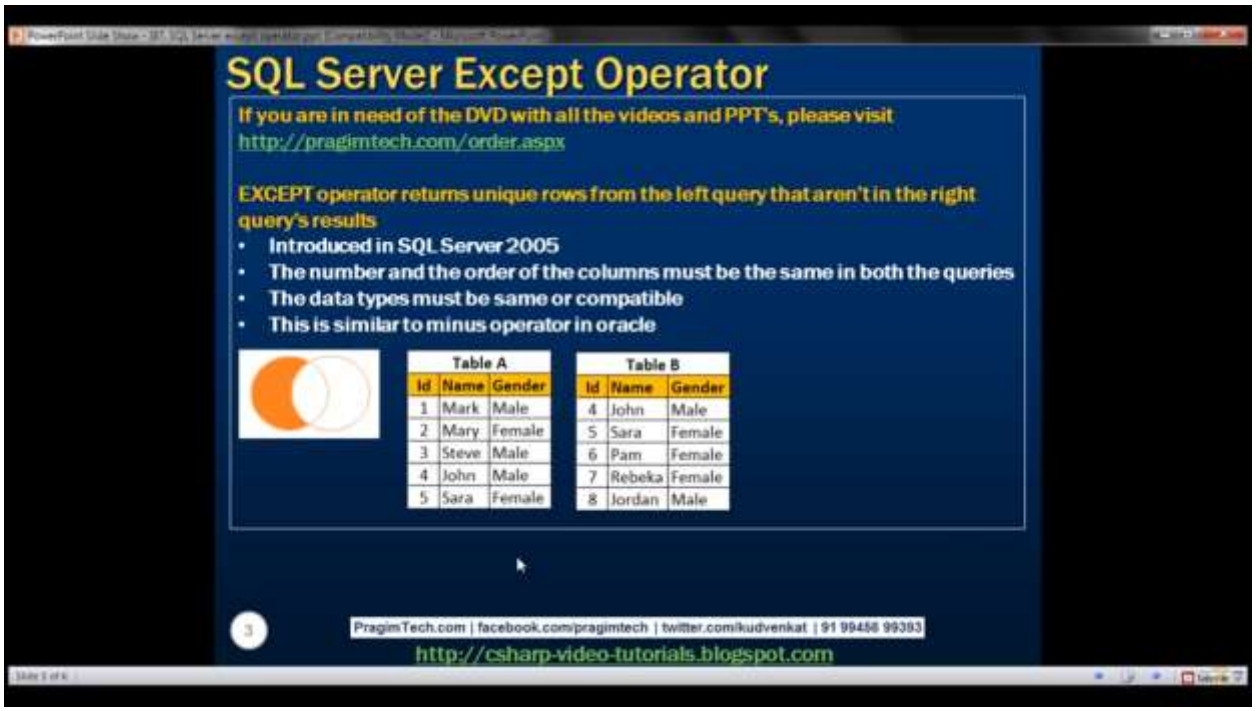

Select * from tblProducts
Select * from tblProductSales
Select Id, Name, [Description]
from tblProducts
where Id NOT IN (Select distinct ProductId from tblProductSales)
Select tblProducts.Id, Name, [Description]
from tblProducts
left join tblProductSales
on tblProducts.Id = tblProductSales.ProductId
where tblProductSales.ProductId is null

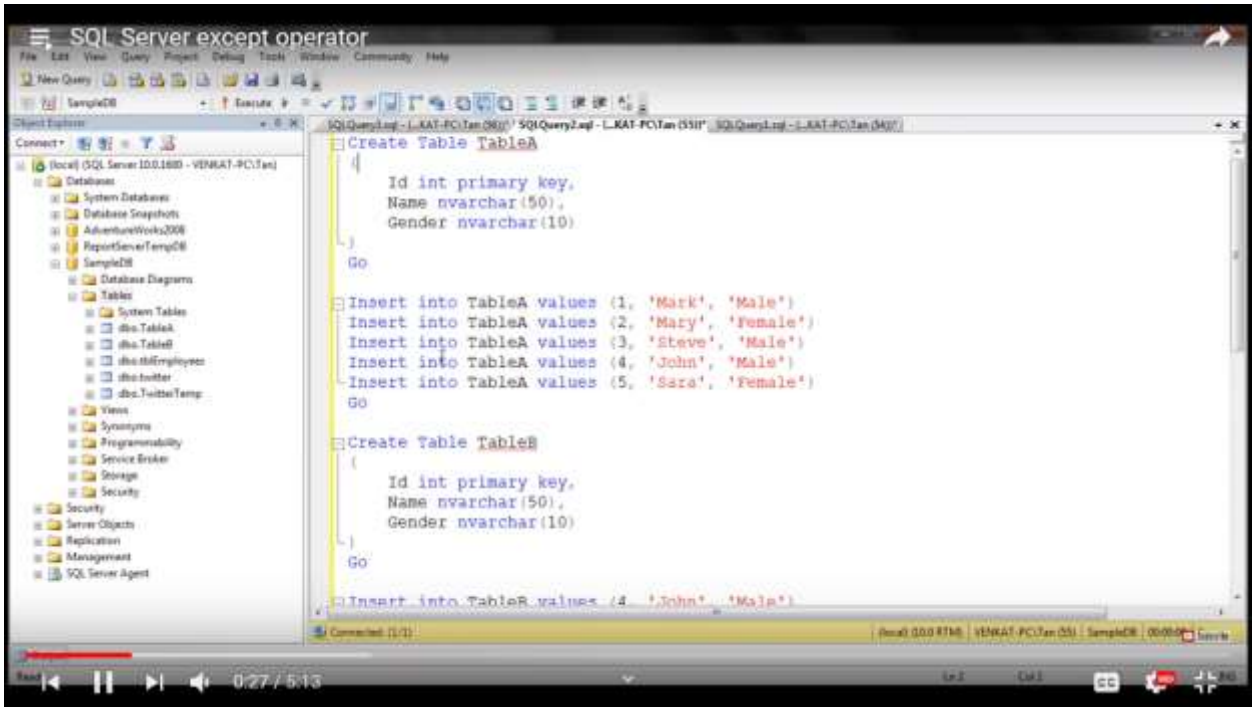

Correlated subquery
If the subquery depends on the outer query for its values, then that sub query is called as correlated subquery.
In the where clause of the subquery below, "ProductId" column get it's value from tblProducts table that is present in the outer query.
Select [Name],
(Select SUM(QuantitySold) from tblProductSales
where ProductId = tblProducts.Id) as TotalQuantity
from tblProducts
order by Name
So, here the subquery is dependent on the outer query for it's value, hence this subquery is a correlated subquery.
Correlated subqueries get executed, once for every row that is selected by the outer query.
Corelated subquery, cannot be executed independently of the outer query.
3
PRAGIM Technologies | 9000113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com

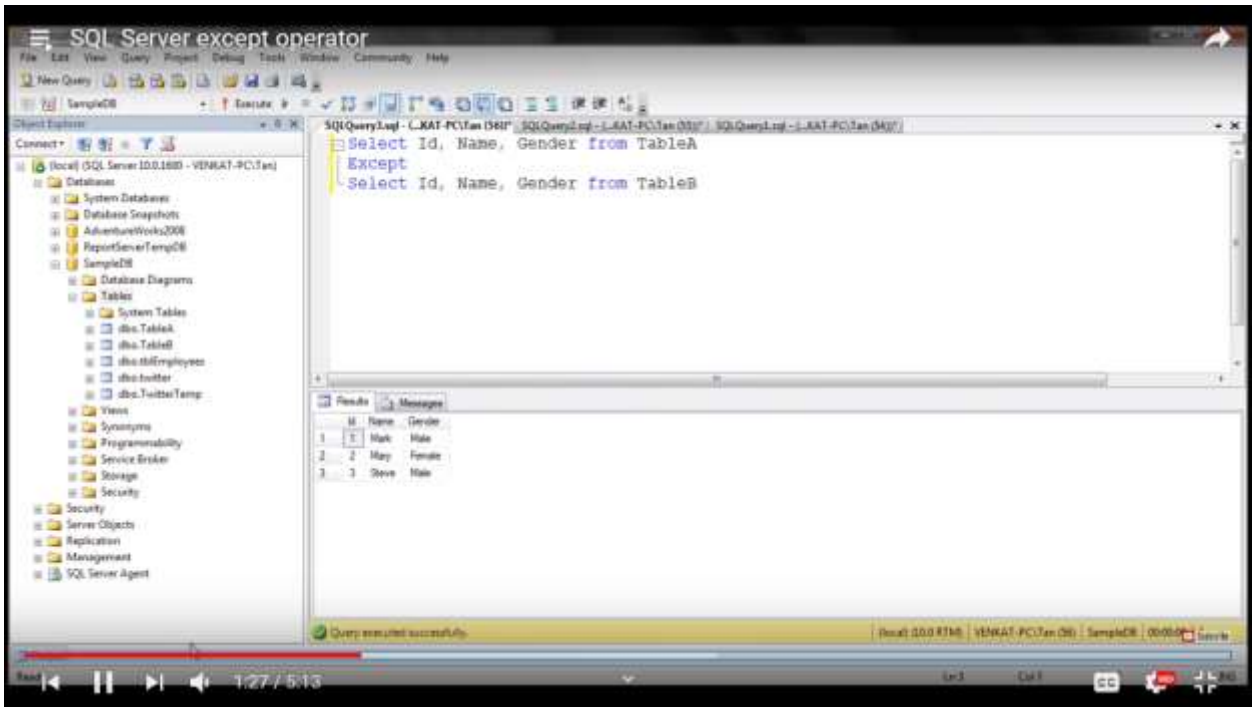

What to choose for performance SubQuery or Joins Part 62
Performance - SubQueries or Joins
According to MSDN, in most cases, there is usually no performance difference between queries that uses sub-queries and equivalent queries using joins.
According to MSDN, in some cases where existence must be checked, a join produces better performance. Otherwise, the nested query must be processed for each result of the outer query. In such cases, a join approach would yield better results.
In general joins work faster than sub-queries, but in reality it all depends on the execution plan that is generated by SQL Server. Id does not matter how we have written the query, SQL Server will always transform it on an execution plan. If it is "smart" enough to generate the same plan from both queries, you will get the same result.
I would say, rather than going by theory, turn on client statistics and execution plan to see the performance of each option, and then make a decision. In a later video session we will discuss about client statistics and execution plans in detail.
Ensure your Child's Future is Secured, even in your absence.
Buy ₹50 LAKH TERM INSURANCE PLAN @ ₹372 P.M.*
http://csharp-video-tutorials.blogspot.com
0:51 / 12:25

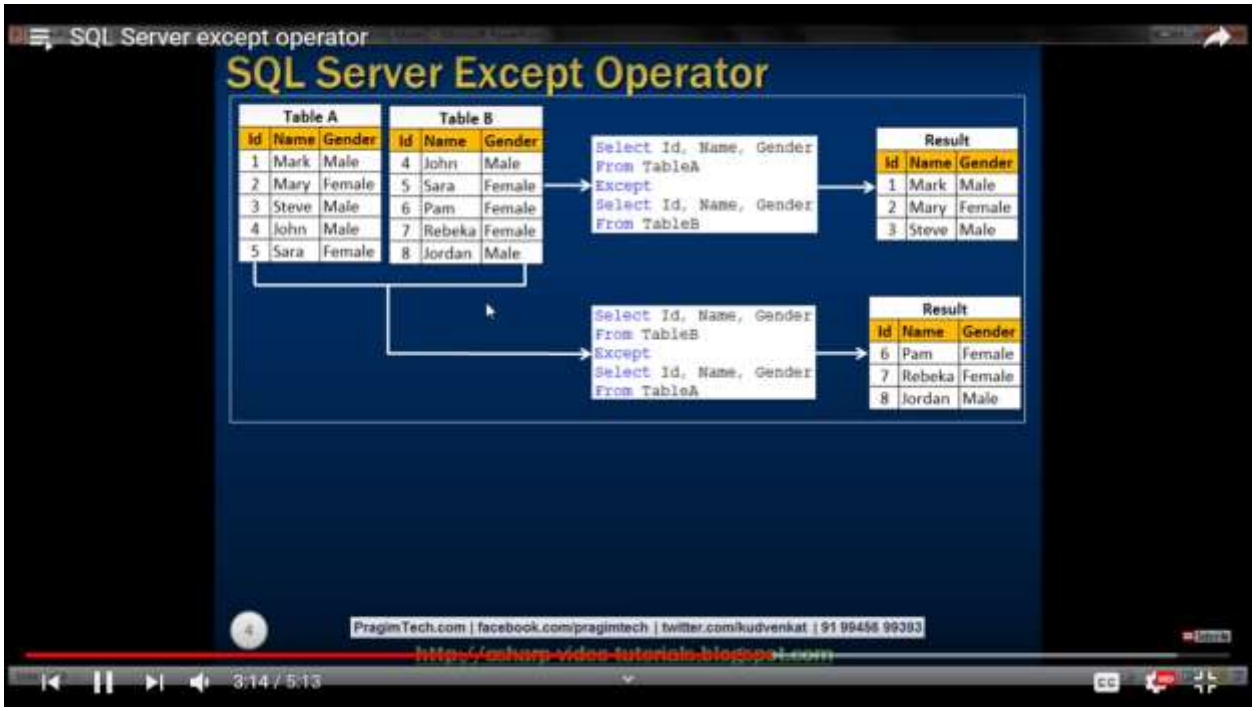

Cursors in sql server Part 63
Cursors in SQL Server
Relational Database Management Systems, including sql server are very good at handling data in SETS. For example, the following "UPDATE" query, updates a set of rows that matches the condition in the "WHERE" clause at the same time.
Update tblProductSales Set UnitPrice = 50 where ProductId = 101
tblProducts
Id Name Description
1 Product - 1 Product - 1 Description
2 Product - 2 Product - 2 Description
3 Product - 3 Product - 3 Description
4 Product - 4 Product - 4 Description
5 Product - 5 Product - 5 Description
tblProductSales
Id ProductId UnitPrice QuantitySold
1 5 5 3
2 4 23 4
3 3 31 2
4 4 93 9
5 5 72 5
However, if there is ever a need to process the rows, on a row-by-row basis, then cursors are your choice. Cursors are very bad for performance, and should be avoided always. Most of the time, cursors can be very easily replaced using joins.
PRAGIM Technologies | 9900113931 | www.pragimtech.com | www.facebook.com/pragimtech
http://csharp-video-tutorials.blogspot.com
0:21 / 13:57

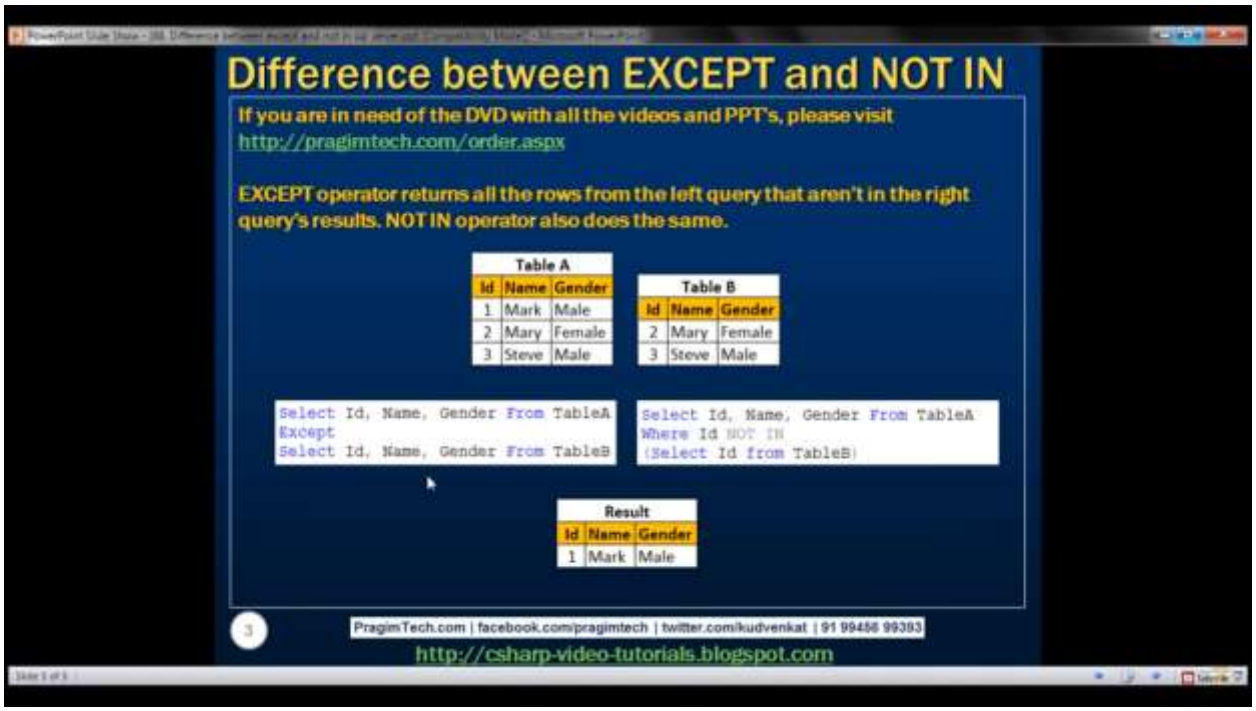

Cursors in sql server Part 63
Declare @ProductId int
Declare @Name nvarchar(30)
Declare ProductCursor CURSOR FOR
Select Id,Name from tblProducts where Id <= 1000
Open ProductCursor
Fetch Next from ProductCursor into @ProductId, @Name
While(@@FETCH_STATUS = 0)
Begin
Fetch Next from ProductCursor into @ProductId, @Name
End
CLOSE ProductCursor
Exit full screen (f)
4:04 / 13:57

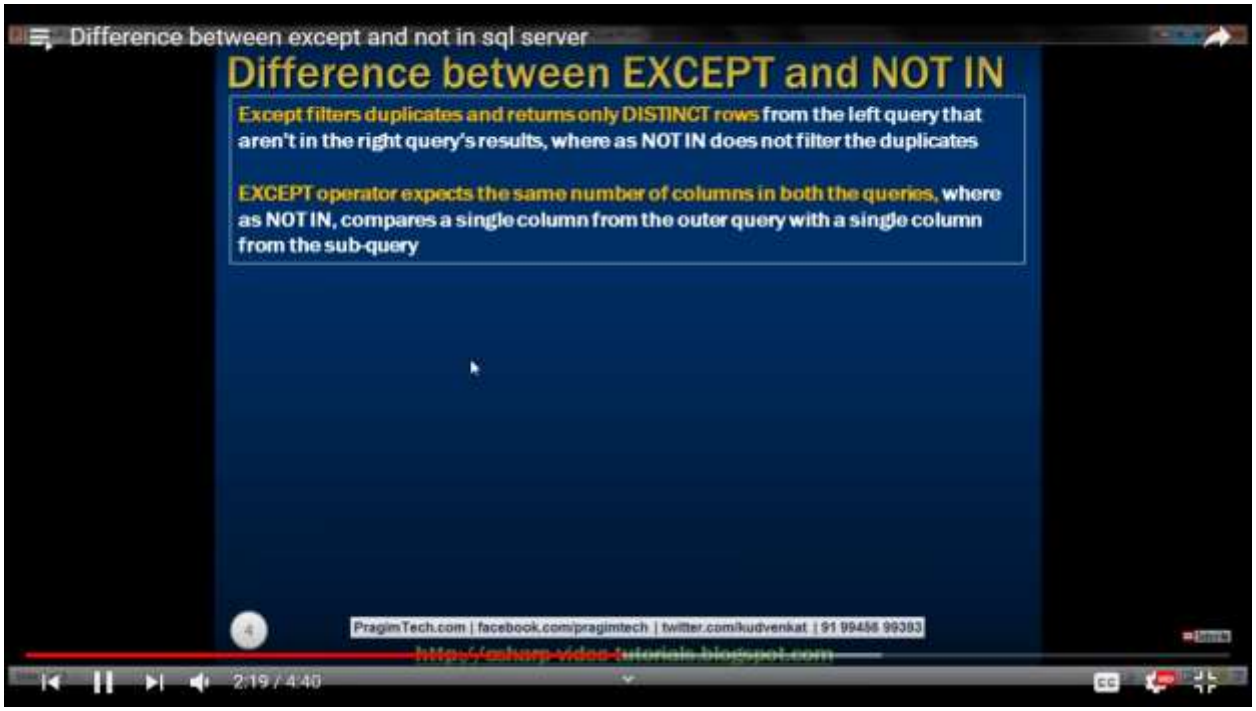

SQL Server Except Operator
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
EXCEPT operator returns unique rows from the left query that aren't in the right query's results
Introduced in SQL Server 2005
The number and the order of the columns must be the same in both the queries
The data types must be same or compatible
This is similar to minus operator in oracle
Table A
Id Name Gender
1 Mark Male
2 Mary Female
3 Steve Male
4 John Male
5 Sara Female
Table B
Id Name Gender
4 John Male
5 Sara Female
6 Pam Female
7 Rebeka Female
8 Jordan Male
3
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99393
http://csharp-video-tutorials.blogspot.com

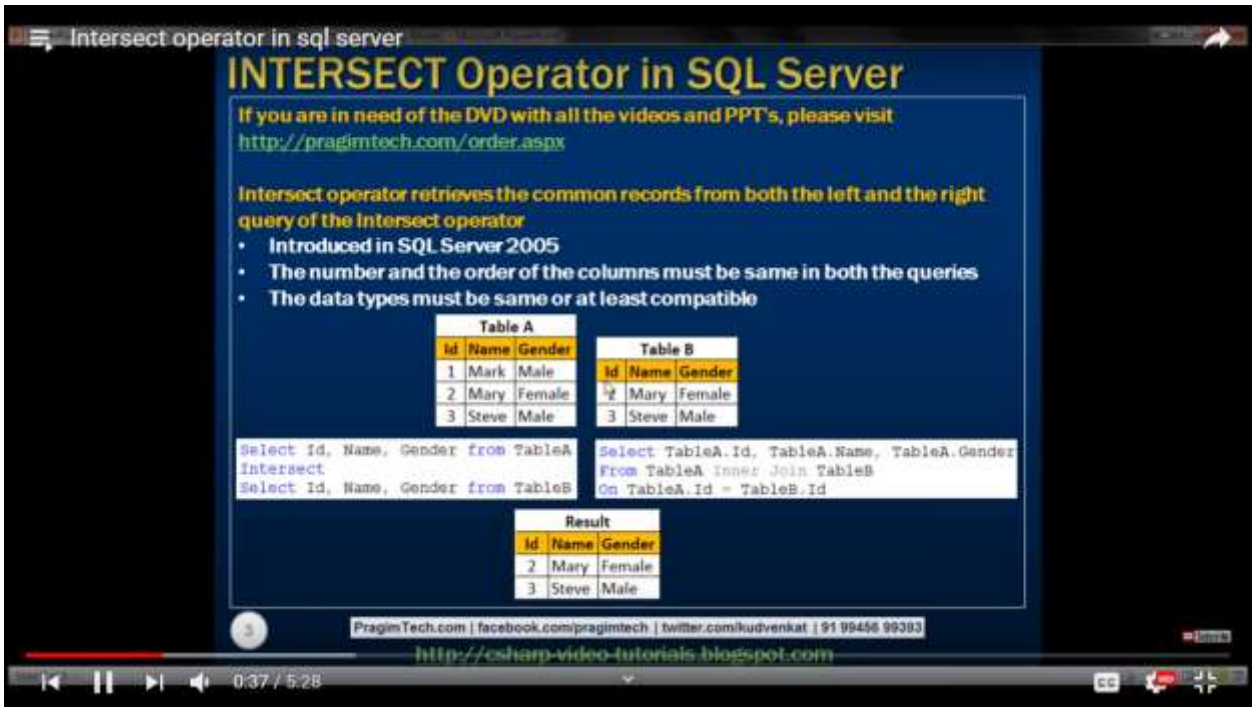

SQL Server except operator
Create Table TableA
(
Id int primary key,
Name nvarchar(50),
Gender nvarchar(10)
)
Go
Insert into TableA values (1, 'Mark', 'Male')
Insert into TableA values (2, 'Mary', 'Female')
Insert into TableA values (3, 'Steve', 'Male')
Insert into TableA values (4, 'John', 'Male')
Insert into TableA values (5, 'Sara', 'Female')
Go
Create Table TableB
(
Id int primary key,
Name nvarchar(50),
Gender nvarchar(10)
)
Go
0:27 / 5:13

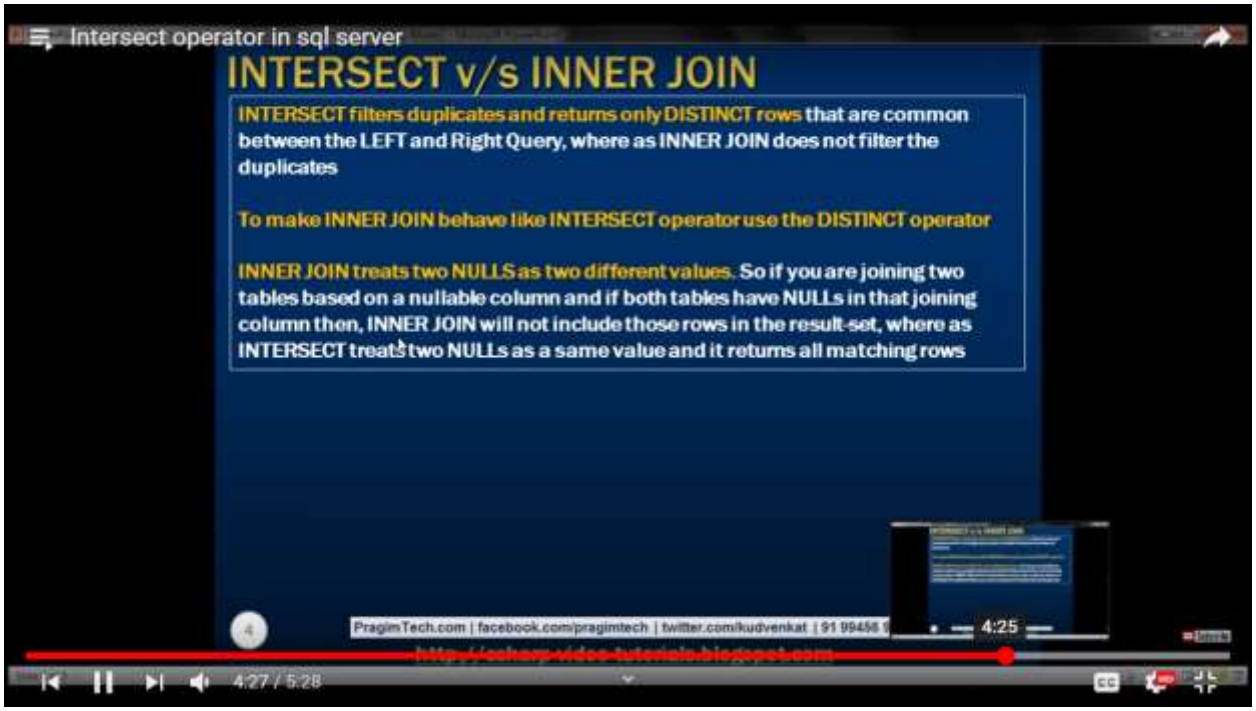

SQL Server except operator
Select Id, Name, Gender from TableA
Except
Select Id, Name, Gender from TableB
1:27 / 5:13

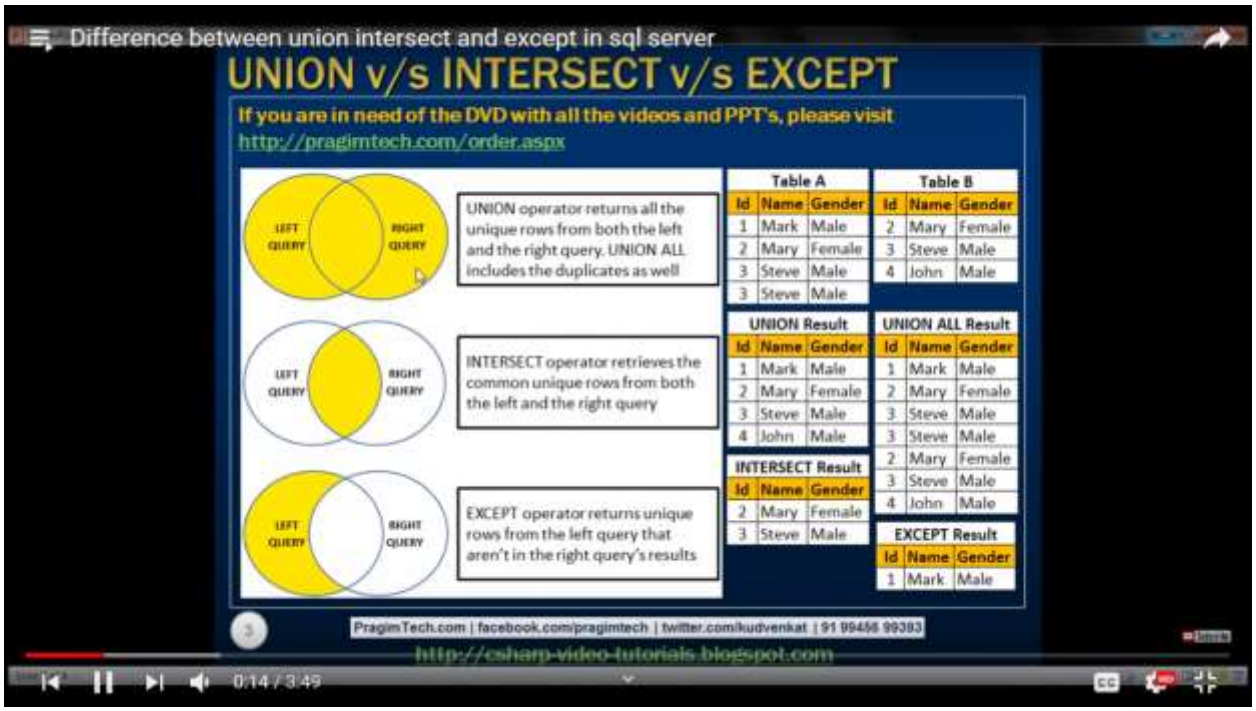

SQL Server except operator
SQL Server Except Operator
Table A
Id Name Gender
1 Mark Male
2 Mary Female
3 Steve Male
4 John Male
5 Sara Female
Table B
Id Name Gender
4 John Male
5 Sara Female
6 Pam Female
7 Rebeka Female
8 Jordan Male
Select Id, Name, Gender
From TableA
Except
Select Id, Name, Gender
From TableB
Result
Id Name Gender
1 Mark Male
2 Mary Female
3 Steve Male
Select Id, Name, Gender
From TableB
Except
Select Id, Name, Gender
From TableA
Result
Id Name Gender
6 Pam Female
7 Rebeka Female
8 Jordan Male
4
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99458 99393
http://csharp-video-tutorials.blogspot.com
3:14 / 5:13

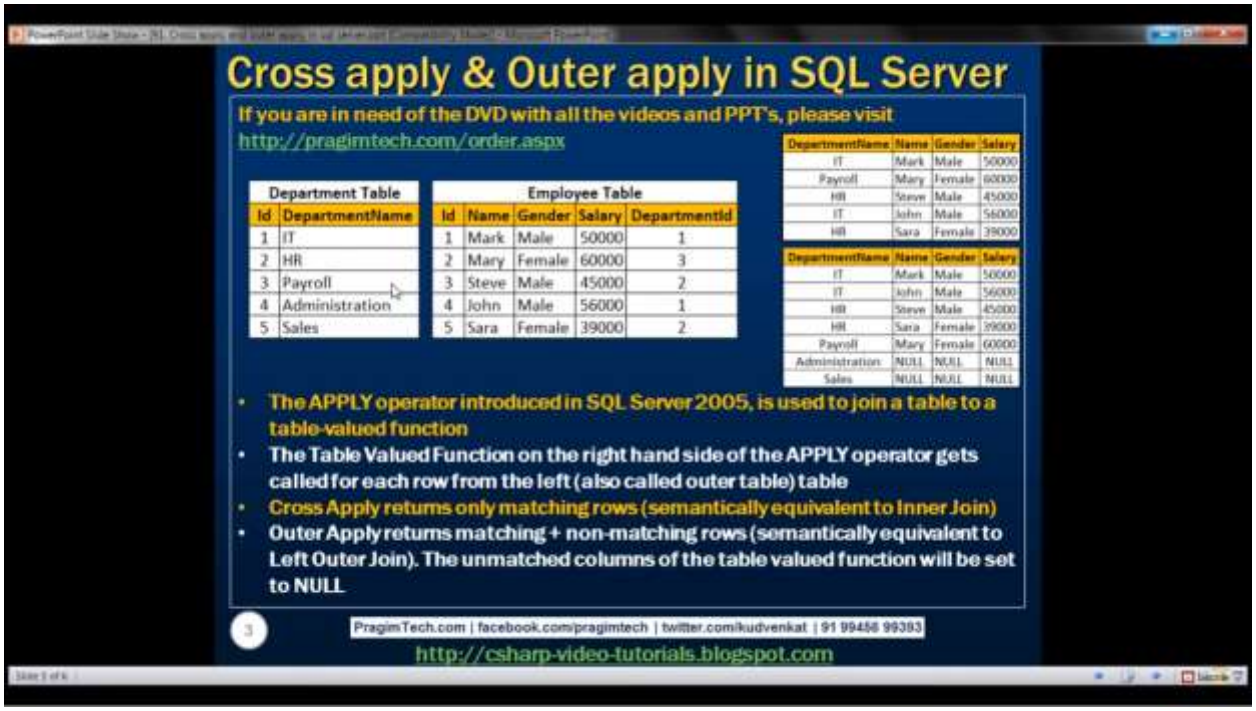

Difference between EXCEPT and NOT IN
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
EXCEPT operator returns all the rows from the left query that aren't in the right query's results. NOT IN operator also does the same.
Table A
Id Name Gender
1 Mark Male
2 Mary Female
3 Steve Male
Table B
Id Name Gender
2 Mary Female
3 Steve Male
Select Id, Name, Gender From TableA
Except
Select Id, Name, Gender From TableB
Select Id, Name, Gender From TableA
Where Id NOT IN
(Select Id from TableB)
Result
Id Name Gender
1 Mark Male
3
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99458 99393
http://csharp-video-tutorials.blogspot.com

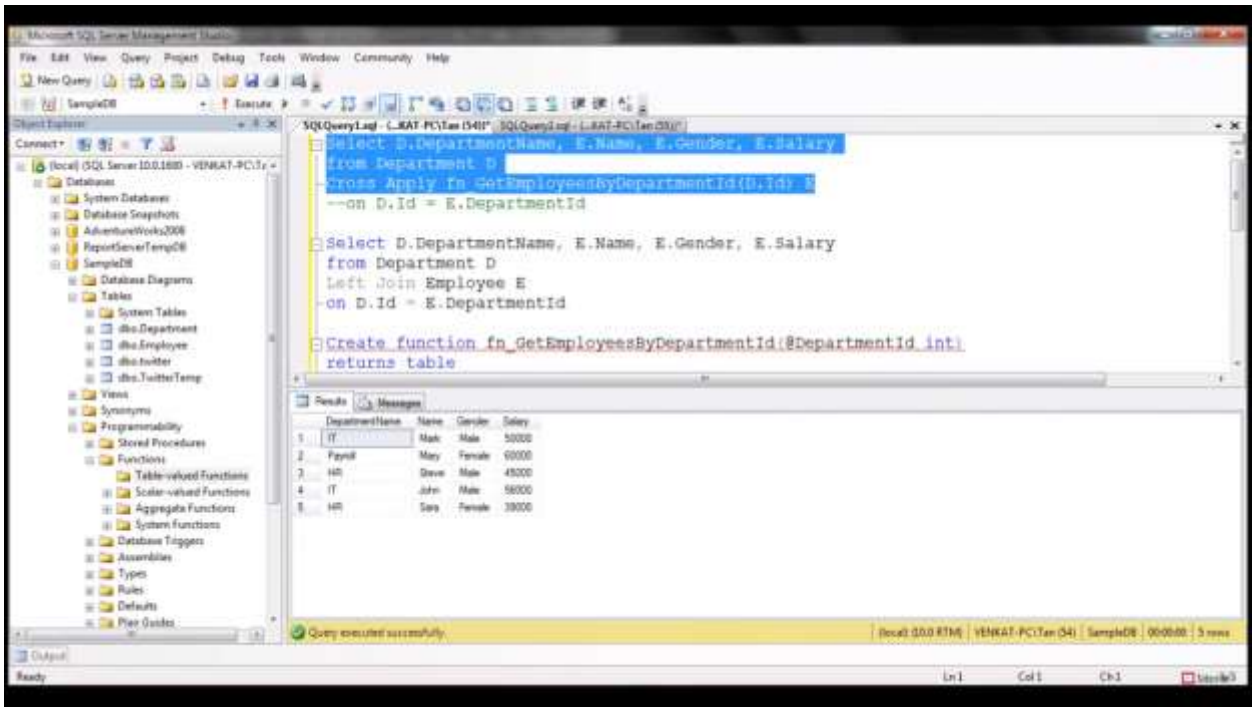

Difference between except and not in sql server
Difference between EXCEPT and NOT IN
Except filters duplicates and returns only DISTINCT rows from the left query that aren't in the right query's results, where as NOT IN does not filter the duplicates
EXCEPT operator expects the same number of columns in both the queries, where as NOT IN, compares a single column from the outer query with a single column from the sub-query
4
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99383
http://csharp-video-tutorials.blogspot.com
2:19 / 4:40

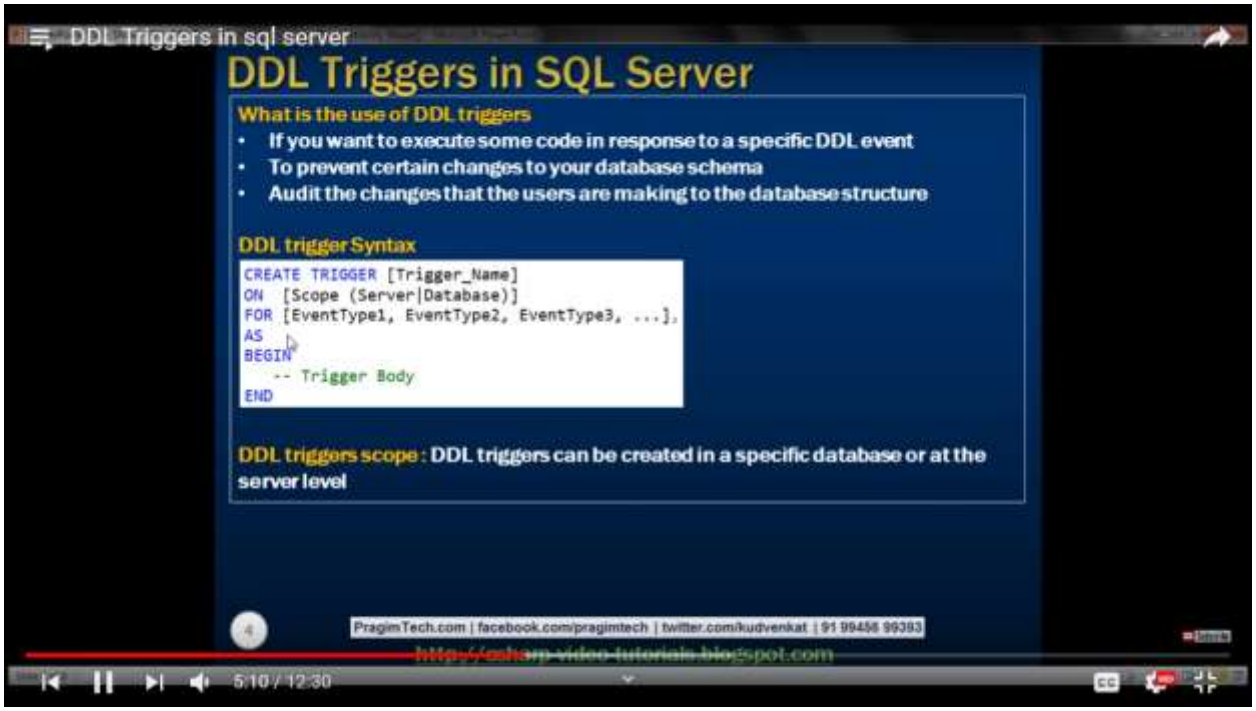

Intersect operator in sql server
INTERSECT Operator in SQL Server
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
Intersect operator retrieves the common records from both the left and the right query of the Intersect operator
• Introduced in SQL Server 2005
• The number and the order of the columns must be same in both the queries
• The data types must be same or at least compatible
Table A
Id Name Gender
1 Mark Male
2 Mary Female
3 Steve Male
Table B
Id Name Gender
2 Mary Female
3 Steve Male
Select Id, Name, Gender from TableA
Intersect
Select Id, Name, Gender from TableB
Select TableA.Id, TableA.Name, TableA.Gender
From TableA Inner Join TableB
On TableA.Id = TableB.Id
Result
Id Name Gender
2 Mary Female
3 Steve Male
3
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99383
http://csharp-video-tutorials.blogspot.com
0:37 / 5:28

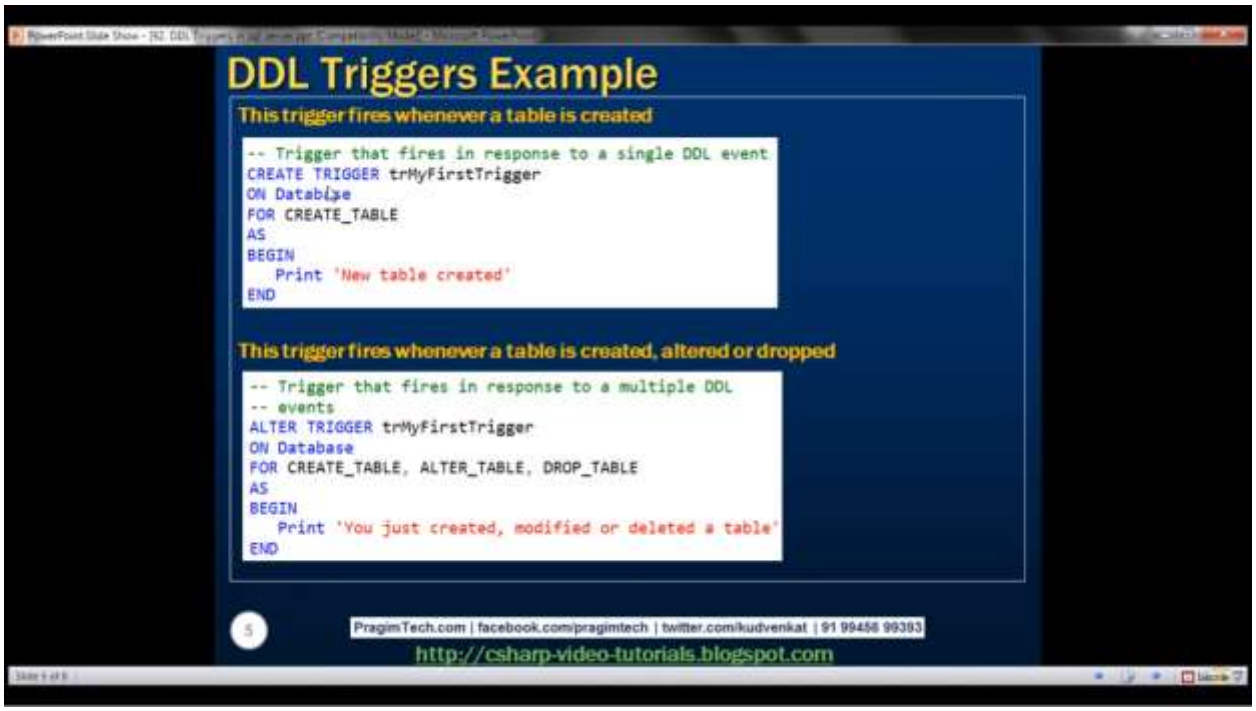

Intersect operator in sql server
INTERSECT v/s INNER JOIN
INTERSECT filters duplicates and returns only DISTINCT rows that are common between the LEFT and Right Query, where as INNER JOIN does not filter the duplicates
To make INNER JOIN behave like INTERSECT operator use the DISTINCT operator
INNER JOIN treats two NULLS as two different values. So if you are joining two tables based on a nullable column and if both tables have NULLs in that joining column then, INNER JOIN will not include those rows in the result-set, where as INTERSECT treats two NULLs as a same value and it returns all matching rows
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456
4:25
4:27 / 5:28

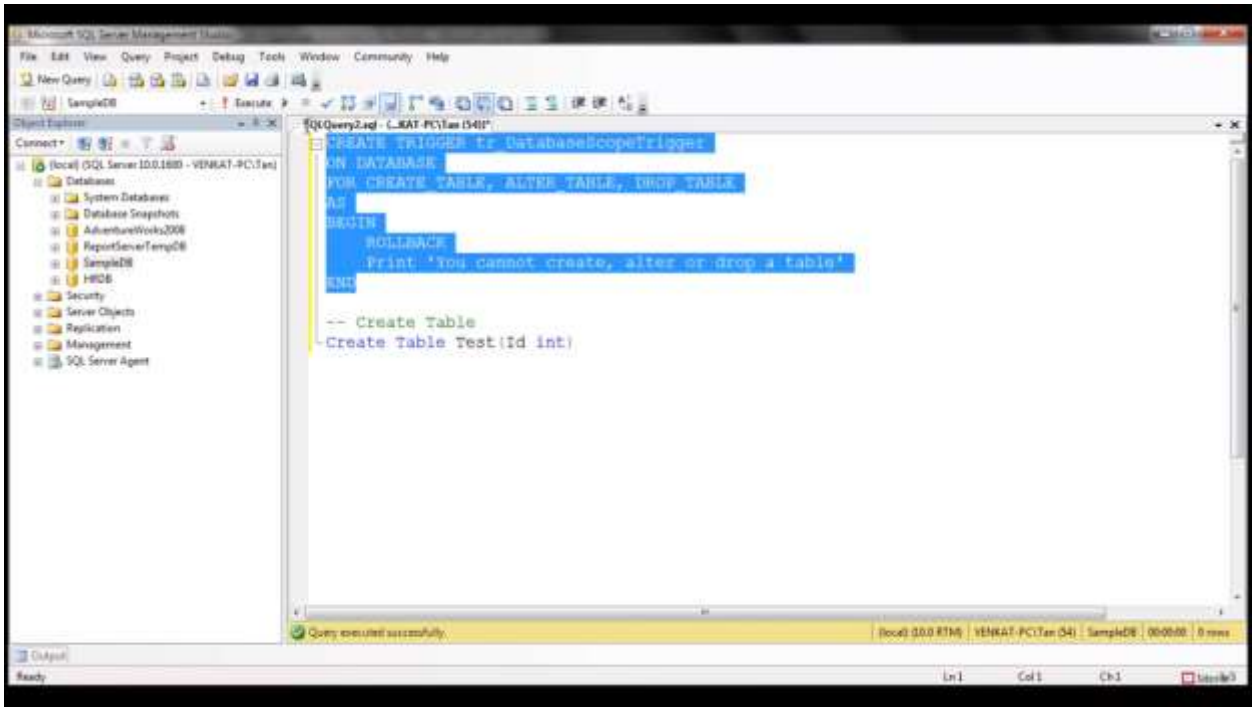

Difference between union intersect and except in sql server
UNION v/s INTERSECT v/s EXCEPT
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
LEFT QUERY
RIGHT QUERY
UNION operator returns all the unique rows from both the left and the right query. UNION ALL includes the duplicates as well
INTERSECT operator retrieves the common unique rows from both the left and the right query
EXCEPT operator returns unique rows from the left query that aren't in the right query's results
Table A
Id Name Gender
1 Mark Male
2 Mary Female
3 Steve Male
3 Steve Male
Table B
Id Name Gender
2 Mary Female
3 Steve Male
4 John Male
UNION Result
Id Name Gender
1 Mark Male
2 Mary Female
3 Steve Male
4 John Male
UNION ALL Result
Id Name Gender
1 Mark Male
2 Mary Female
3 Steve Male
3 Steve Male
2 Mary Female
3 Steve Male
4 John Male
INTERSECT Result
Id Name Gender
2 Mary Female
3 Steve Male
EXCEPT Result
Id Name Gender
1 Mark Male
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99383
0:14 / 3:49

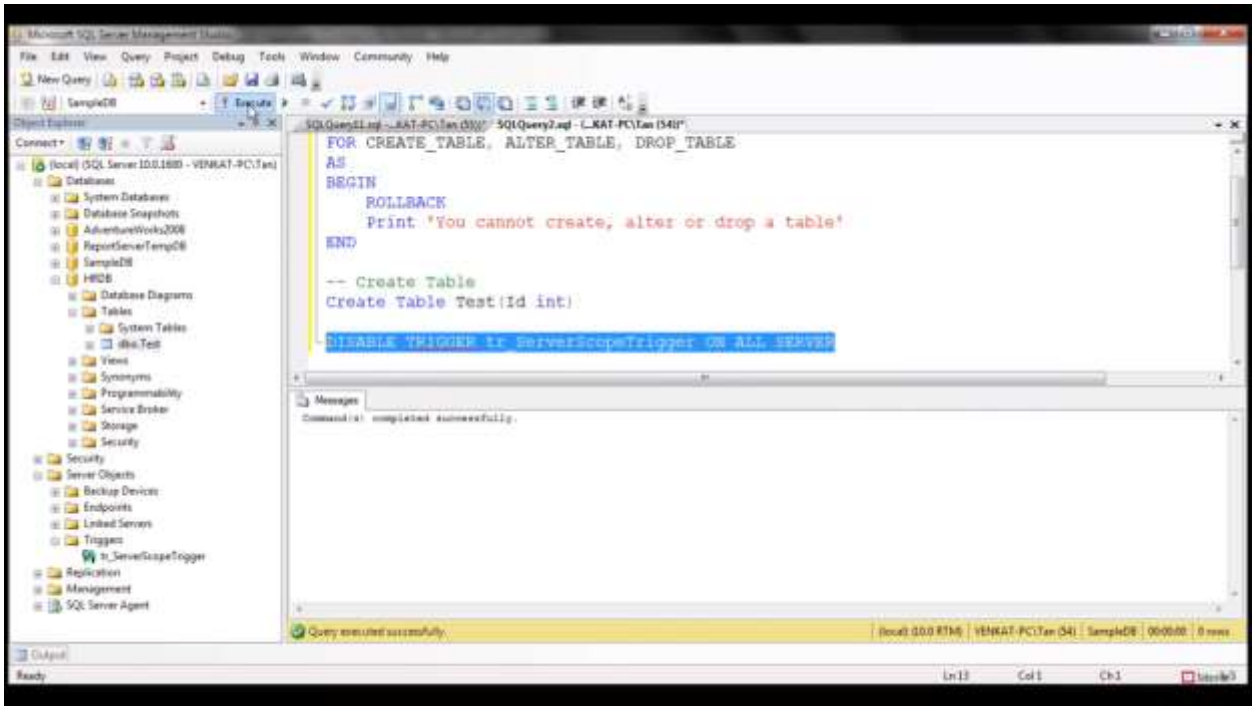

Cross apply & Outer apply in SQL Server
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
Department Table
Id DepartmentName
1 IT
2 HR
3 Payroll
4 Administration
5 Sales
Employee Table
Id Name Gender Salary DepartmentId
1 Mark Male 50000 1
2 Mary Female 60000 3
3 Steve Male 45000 2
4 John Male 56000 1
5 Sara Female 39000 2
DepartmentName Name Gender Salary
IT Mark Male 50000
Payroll Mary Female 60000
HR Steve Male 45000
IT John Male 56000
HR Sara Female 39000
DepartmentName Name Gender Salary
IT Mark Male 50000
IT John Male 56000
HR Steve Male 45000
HR Sara Female 39000
Payroll Mary Female 60000
Administration NULL NULL NULL
Sales NULL NULL NULL
The APPLY operator introduced in SQL Server 2005, is used to join a table to a table-valued function
The Table Valued Function on the right hand side of the APPLY operator gets called for each row from the left (also called outer table) table
Cross Apply returns only matching rows (semantically equivalent to Inner Join)
Outer Apply returns matching + non-matching rows (semantically equivalent to Left Outer Join). The unmatched columns of the table valued function will be set to NULL
3
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99458 99393
http://csharp-video-tutorials.blogspot.com

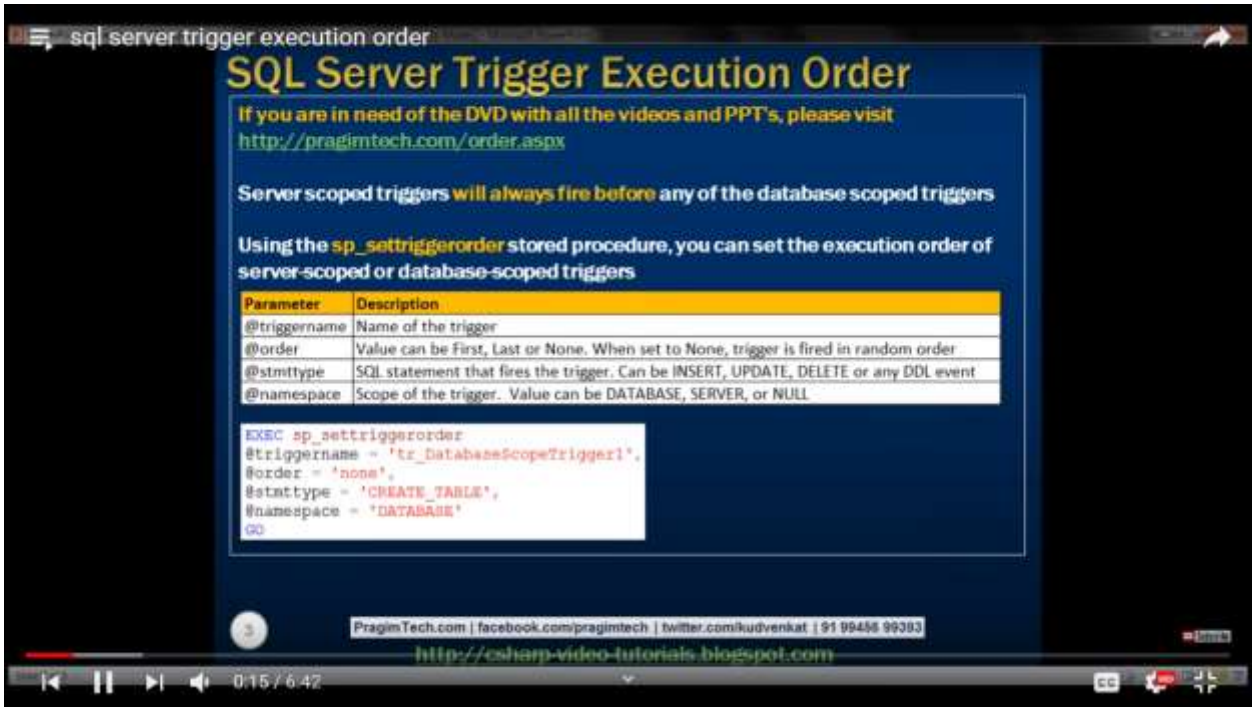

Select D.DepartmentName, E.Name, E.Gender, E.Salary
from Department D
Cross Apply fn_GetEmployeesByDepartmentId(D.Id) E
--on D.Id = E.DepartmentId
Select D.DepartmentName, E.Name, E.Gender, E.Salary
from Department D
Left Join Employee E
on D.Id = E.DepartmentId
Create function fn_GetEmployeesByDepartmentId(@DepartmentId int)
returns table

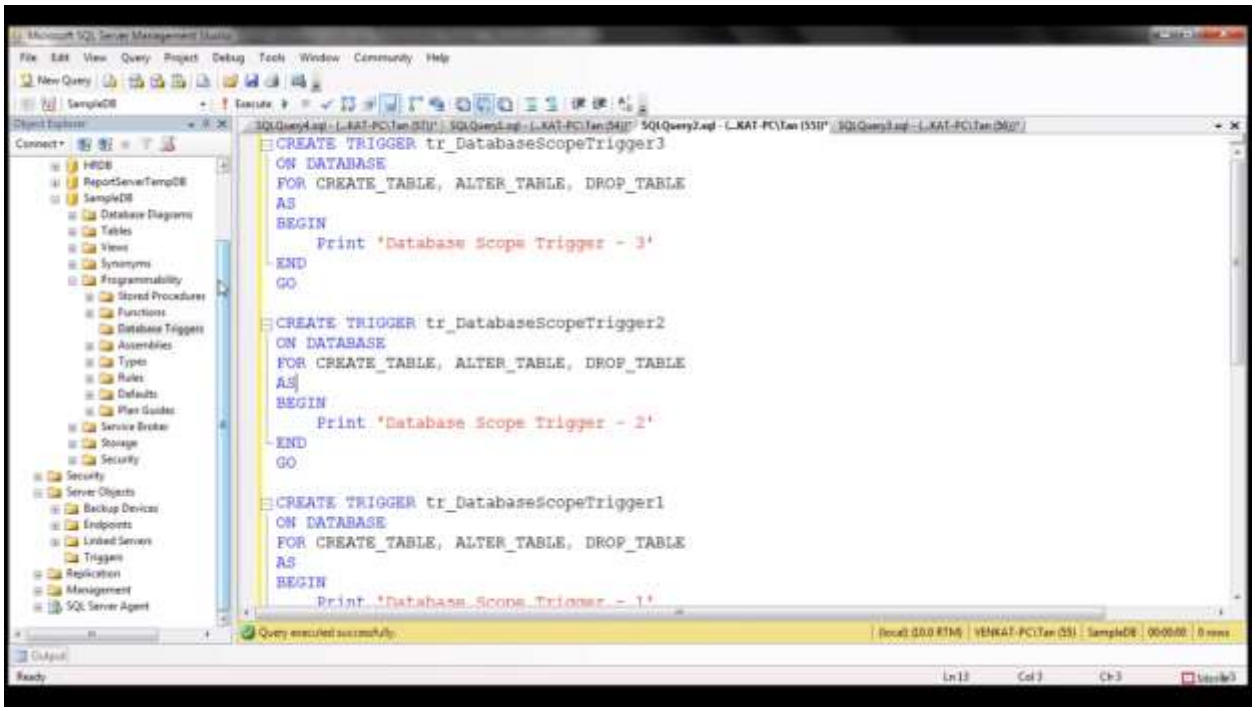

DDL Triggers in sql server
DDL Triggers in SQL Server
What is the use of DDL triggers
If you want to execute some code in response to a specific DDL event
To prevent certain changes to your database schema
Audit the changes that the users are making to the database structure
DDL trigger Syntax
CREATE TRIGGER [Trigger_Name]
ON [Scope (Server|Database)]
FOR [EventType1, EventType2, EventType3, ...],
AS
BEGIN
-- Trigger Body
END
DDL triggers scope : DDL triggers can be created in a specific database or at the server level
4
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99393
http://csharp-video-tutorials.blogspot.com
5:10 / 12:30

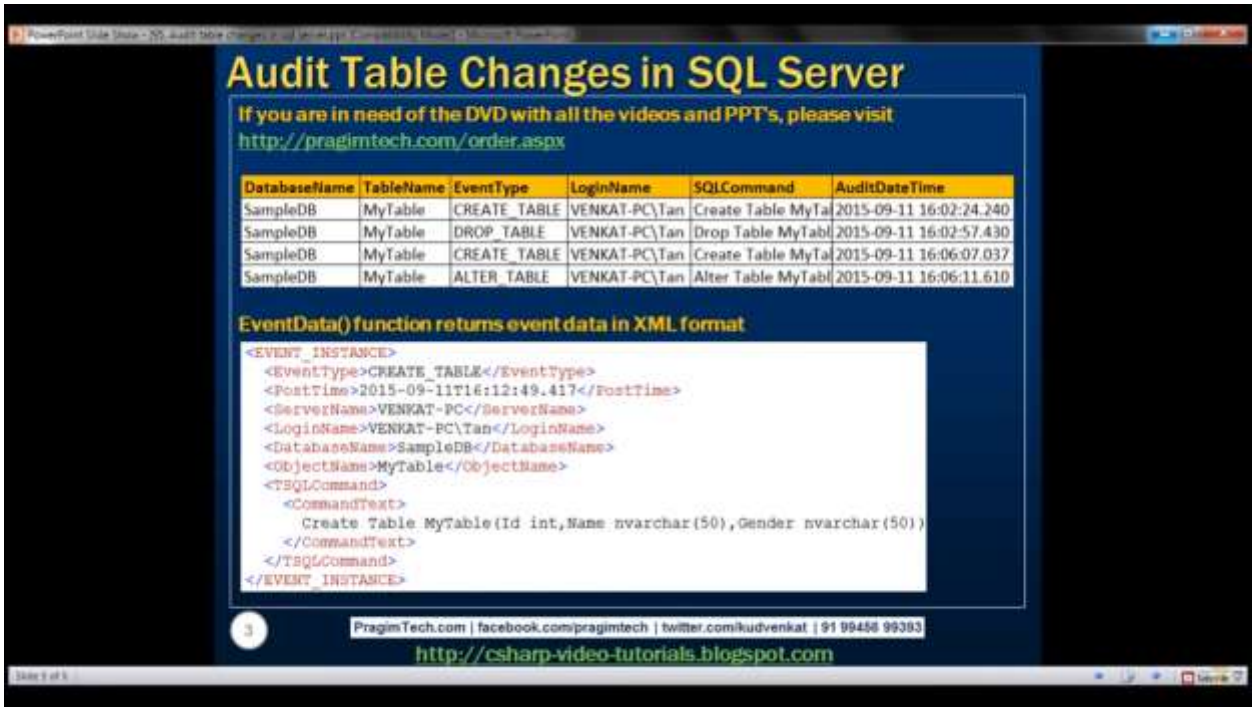

DDL Triggers Example
This trigger fires whenever a table is created
-- Trigger that fires in response to a single DDL event
CREATE TRIGGER trMyFirstTrigger
ON Database
FOR CREATE_TABLE
AS
BEGIN
Print 'New table created'
END
This trigger fires whenever a table is created, altered or dropped
-- Trigger that fires in response to a multiple DDL
-- events
ALTER TRIGGER trMyFirstTrigger
ON Database
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
Print 'You just created, modified or deleted a table'
END
5
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99393
http://csharp-video-tutorials.blogspot.com

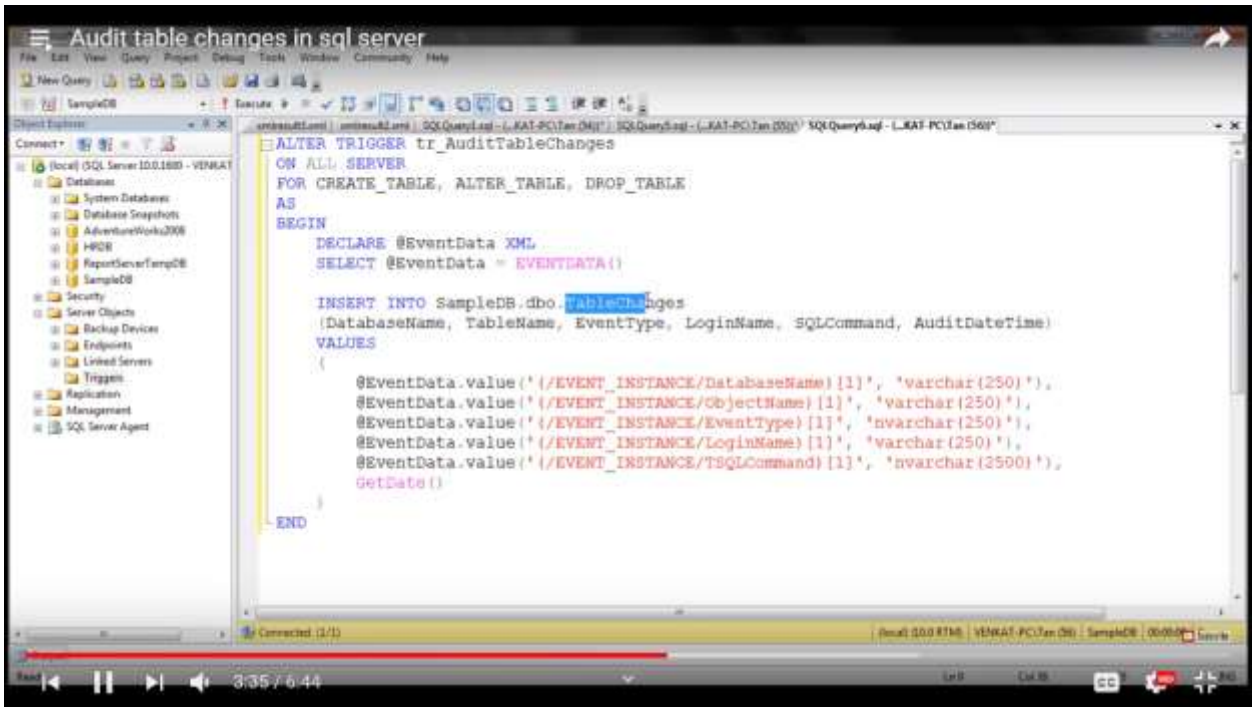

CREATE TRIGGER tr_DatabaseScopeTrigger
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
ROLLBACK
Print 'You cannot create, alter or drop a table'
END
-- Create Table
Create Table Test(Id int)


FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
ROLLBACK
Print 'You cannot create, alter or drop a table'
END
-- Create Table
Create Table Test(Id int)
DISABLE TRIGGER tr_ServerScopeTrigger ON ALL SERVER


sql server trigger execution order
SQL Server Trigger Execution Order
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
Server scoped triggers will always fire before any of the database scoped triggers
Using the sp_settriggerorder stored procedure, you can set the execution order of server-scoped or database-scoped triggers
Parameter
Description
@triggername
Name of the trigger
@order
Value can be First, Last or None. When set to None, trigger is fired in random order
@stmttype
SQL statement that fires the trigger. Can be INSERT, UPDATE, DELETE or any DDL event
@namespace
Scope of the trigger. Value can be DATABASE, SERVER, or NULL
EXEC sp_settriggerorder
@triggername = 'tr_DatabaseScopeTrigger1',
@order = 'none',
@stmttype = 'CREATE_TABLE',
@namespace = 'DATABASE'
GO
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99458 99383
http://csharp-video-tutorials.blogspot.com
0:15 / 6:42


CREATE TRIGGER tr_DatabaseScopeTrigger3
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
Print 'Database Scope Trigger - 3'
END
GO
CREATE TRIGGER tr_DatabaseScopeTrigger2
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
Print 'Database Scope Trigger - 2'
END
GO
CREATE TRIGGER tr_DatabaseScopeTrigger1
ON DATABASE
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
Print 'Database Scope Trigger - 1'


Audit Table Changes in SQL Server
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
DatabaseName | TableName | EventType | LoginName | SQLCommand | AuditDateTime
SampleDB | MyTable | CREATE_TABLE | VENKAT-PC\Tan | Create Table MyTal | 2015-09-11 16:02:24.240
SampleDB | MyTable | DROP_TABLE | VENKAT-PC\Tan | Drop Table MyTabl | 2015-09-11 16:02:57.430
SampleDB | MyTable | CREATE_TABLE | VENKAT-PC\Tan | Create Table MyTal | 2015-09-11 16:06:07.037
SampleDB | MyTable | ALTER_TABLE | VENKAT-PC\Tan | Alter Table MyTabl | 2015-09-11 16:06:11.610
EventData() function returns event data in XML format
<EVENT_INSTANCE>
<EventType>CREATE_TABLE</EventType>
<PostTime>2015-09-11T16:12:49.417</PostTime>
<ServerName>VENKAT-PC</ServerName>
<LoginName>VENKAT-PC\Tan</LoginName>
<DatabaseName>SampleDB</DatabaseName>
<ObjectName>MyTable</ObjectName>
<TSQLCommand>
<CommandText>
Create Table MyTable(Id int,Name nvarchar(50),Gender nvarchar(50))
</CommandText>
</TSQLCommand>
</EVENT_INSTANCE>
3
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99458 99393
http://csharp-video-tutorials.blogspot.com


Audit table changes in sql server
ALTER TRIGGER tr_AuditTableChanges
ON ALL SERVER
FOR CREATE_TABLE, ALTER_TABLE, DROP_TABLE
AS
BEGIN
DECLARE @EventData XML
SELECT @EventData = EVENTDATA()
INSERT INTO SampleDB.dbo.TableChanges
(DatabaseName, TableName, EventType, LoginName, SQLCommand, AuditDateTime)
VALUES
(
@EventData.value('(/EVENT_INSTANCE/DatabaseName)[1]', 'varchar(250)'),
@EventData.value('(/EVENT_INSTANCE/ObjectName)[1]', 'varchar(250)'),
@EventData.value('(/EVENT_INSTANCE/EventType)[1]', 'nvarchar(250)'),
@EventData.value('(/EVENT_INSTANCE/LoginName)[1]', 'varchar(250)'),
@EventData.value('(/EVENT_INSTANCE/TSQLCommand)[1]', 'nvarchar(2500)'),
GetDate()
)
END
3:35 / 6:44

Logon Triggers in SQL Server

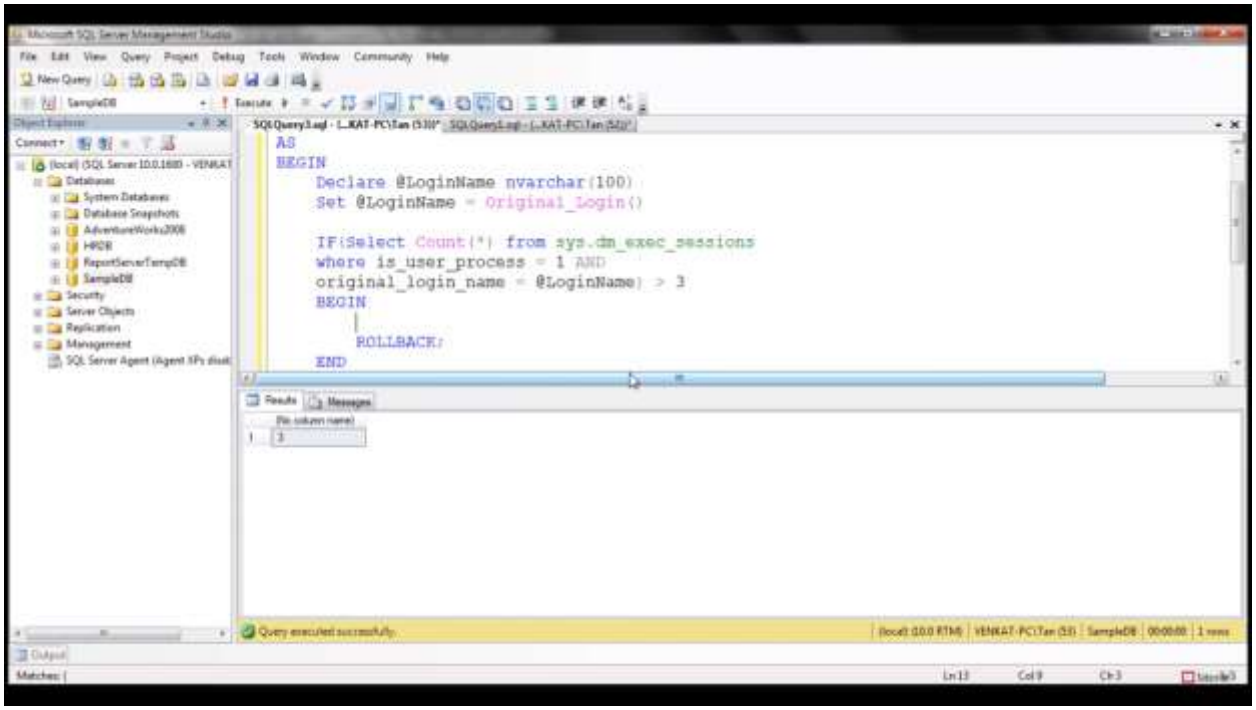
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx

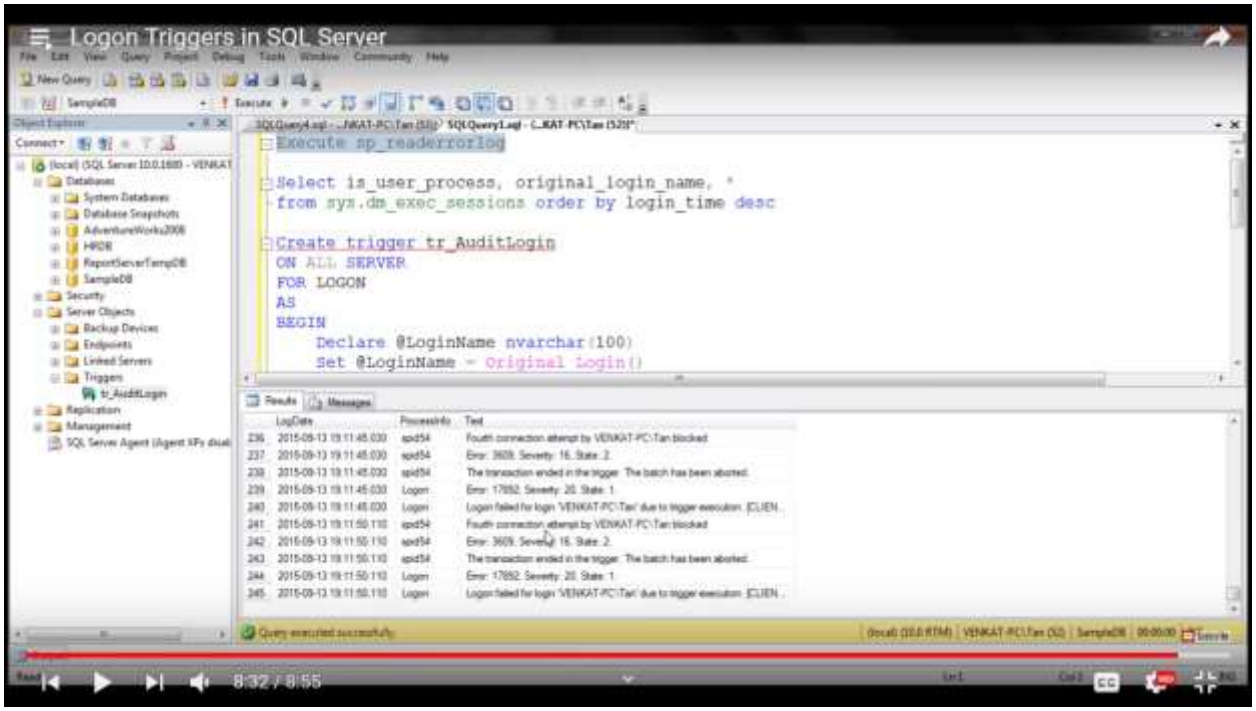
As the name implies Logon triggers fire in response to a LOGON event. Logon triggers fire after the authentication phase of logging in finishes, but before the user session is actually established

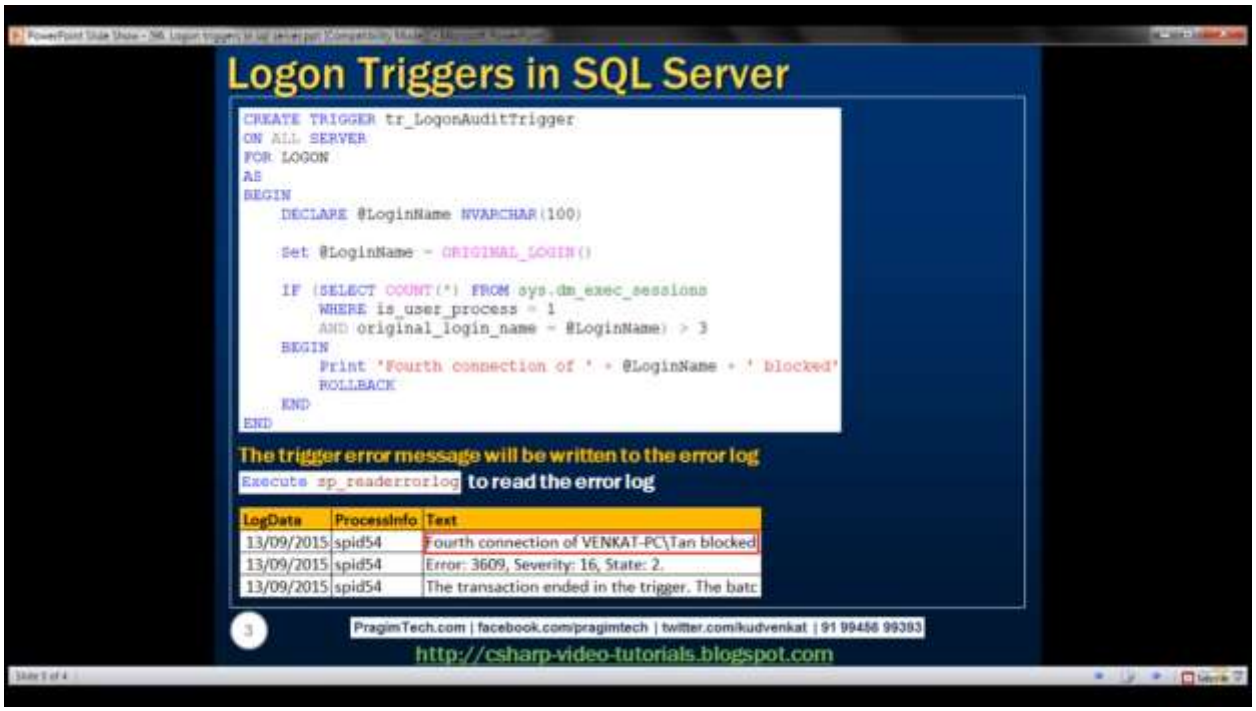
Logon triggers can be used for

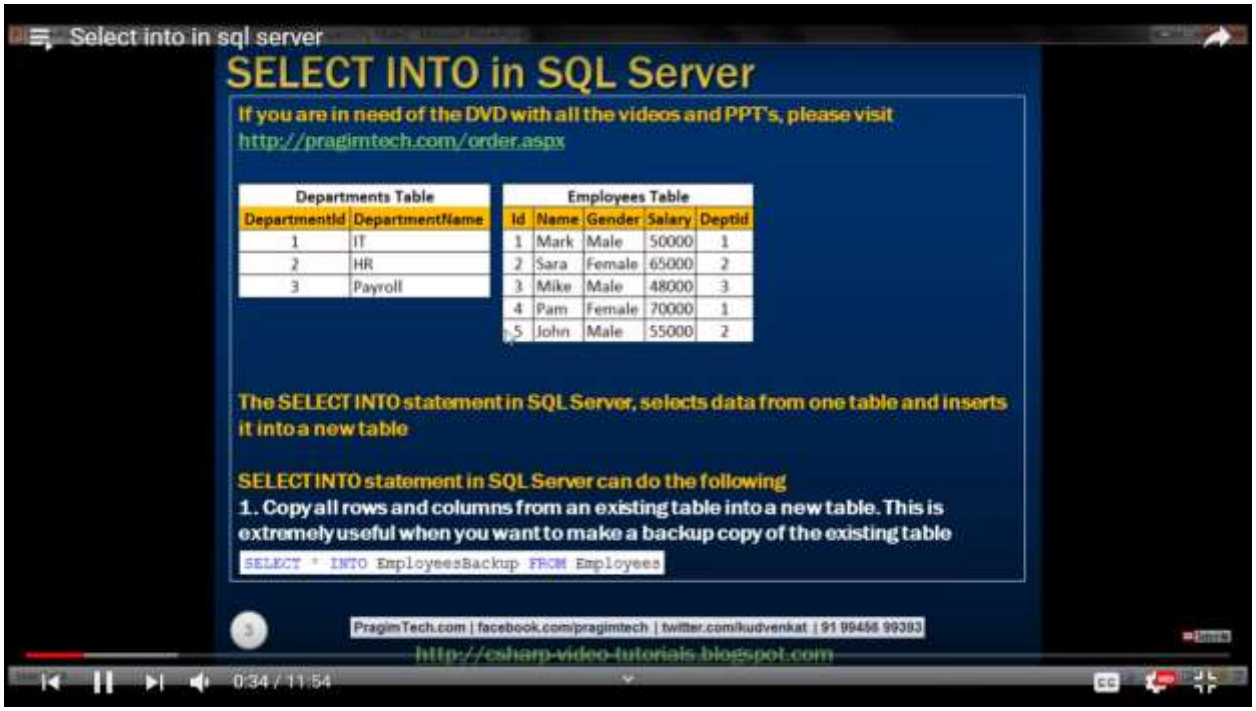
1. Tracking login activity
2. Restricting logins to SQL Server
3. Limiting the number of sessions for a specific login

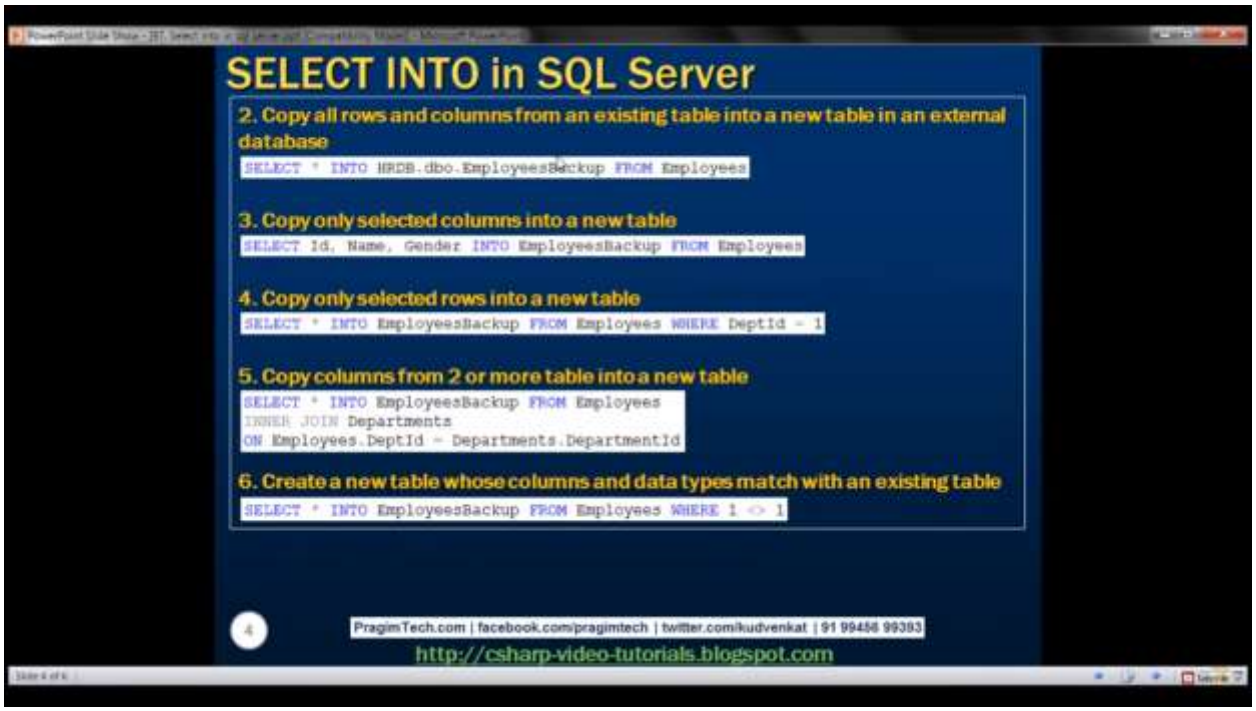
Write a trigger to limit the maximum number of open connections for a user to 3

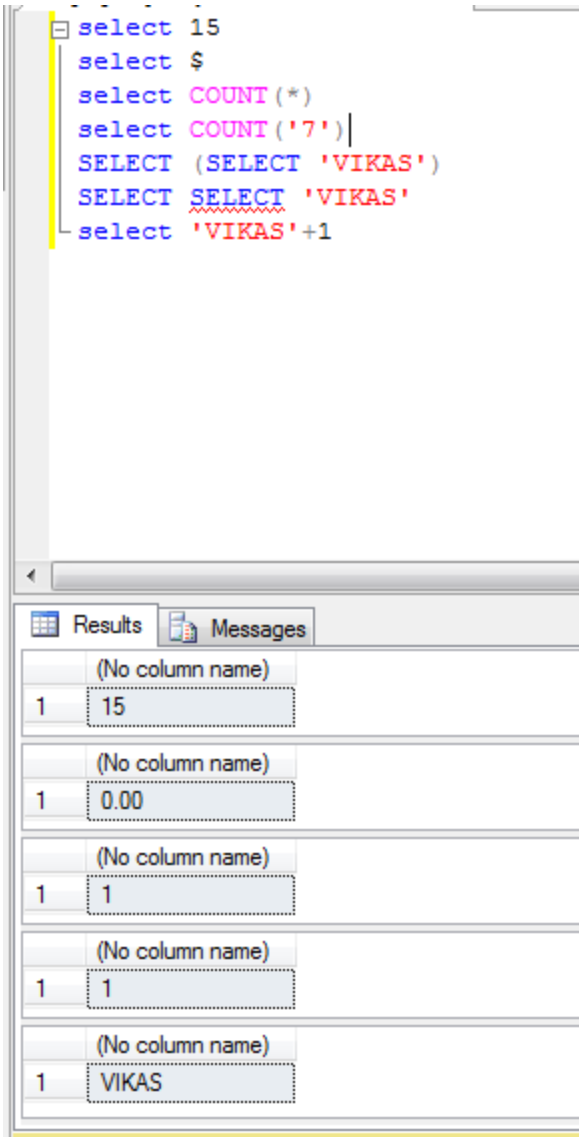



Logon Triggers in SQL Server
Execute sp_readerrorlog
Select is_user_process, original_login_name, *
from sys.dm_exec_sessions order by login_time desc
Create trigger tr_AuditLogin
ON ALL SERVER
FOR LOGON
AS
BEGIN
Declare @LoginName nvarchar(100)
Set @LoginName = Original_Login()
8:32 / 8:55


Logon Triggers in SQL Server
CREATE TRIGGER tr_LogonAuditTrigger
ON ALL SERVER
FOR LOGON
AS
BEGIN
DECLARE @LoginName NVARCHAR(100)
Set @LoginName = ORIGINAL_LOGIN()
IF (SELECT COUNT(*) FROM sys.dm_exec_sessions
WHERE is_user_process = 1
AND original_login_name = @LoginName) > 3
BEGIN
Print 'Fourth connection of ' + @LoginName + ' blocked'
ROLLBACK
END
END
The trigger error message will be written to the error log
Execute sp_readerrorlog to read the error log
LogDate | ProcessInfo | Text
13/09/2015 | spid54 | Fourth connection of VENKAT-PC\Tan blocked
13/09/2015 | spid54 | Error: 3609, Severity: 16, State: 2.
13/09/2015 | spid54 | The transaction ended in the trigger. The batc
3
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99383
http://csharp-video-tutorials.blogspot.com


Select into in sql server
SELECT INTO in SQL Server
If you are in need of the DVD with all the videos and PPT's, please visit
http://pragimtech.com/order.aspx
Departments Table
DepartmentId | DepartmentName
1 | IT
2 | HR
3 | Payroll
Employees Table
Id | Name | Gender | Salary | DeptId
1 | Mark | Male | 50000 | 1
2 | Sara | Female | 65000 | 2
3 | Mike | Male | 48000 | 3
4 | Pam | Female | 70000 | 1
5 | John | Male | 55000 | 2
The SELECT INTO statement in SQL Server, selects data from one table and inserts it into a new table
SELECT INTO statement in SQL Server can do the following
1. Copy all rows and columns from an existing table into a new table. This is extremely useful when you want to make a backup copy of the existing table
SELECT * INTO EmployeesBackup FROM Employees
3
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99393
http://csharp-video-tutorials.blogspot.com
0:34 / 11:54


SELECT INTO in SQL Server
2. Copy all rows and columns from an existing table into a new table in an external database
SELECT * INTO HRDB.dbo.EmployeesBackup FROM Employees
3. Copy only selected columns into a new table
SELECT Id, Name, Gender INTO EmployeesBackup FROM Employees
4. Copy only selected rows into a new table
SELECT * INTO EmployeesBackup FROM Employees WHERE DeptId = 1
5. Copy columns from 2 or more table into a new table
SELECT * INTO EmployeesBackup FROM Employees
INNER JOIN Departments
ON Employees.DeptId = Departments.DepartmentId
6. Create a new table whose columns and data types match with an existing table
SELECT * INTO EmployeesBackup FROM Employees WHERE 1 <> 1
4
PragimTech.com | facebook.com/pragimtech | twitter.com/kudvenkat | 91 99456 99393
http://csharp-video-tutorials.blogspot.com


select 15
select $
select COUNT(*)
select COUNT('7')
SELECT (SELECT 'VIKAS')
SELECT SELECT 'VIKAS'
select 'VIKAS'+1
Results
Messages
(No column name)
1
15
(No column name)
1
0.00
(No column name)
1
1
(No column name)
1
1
(No column name)
1
VIKAS

--101. Write down the query to display all employee name in one cell seprated by ',' ex:-"Vikas, nikita, Ashish, Nikhil , anish" (EmployDetail table)

ANS:

Solution 1:

```
SELECT STUFF(( SELECT ', ' + E.FirstName FROM [EmployeeDetail] AS E FOR XML PATH('')), 1, 2, '') AS [All Emp Name]
```

Output:-

	All Emp Name
1	Vikas, nikita, Ashish, Nikhil , anish

Solution 2:

```
DECLARE @name VARCHAR(MAX) = ''
SELECT  @name = @name + FirstName + ',' FROM  [TMSDB].[dbo].[EmployeeDetails]
SELECT SUBSTRING(@name,1,LEN(@name)-1)
```

100 %

Results | Messages

	(No column name)
1	Vikas,nikita,Ashish,Nikhil,anish

--102. Write down the query to get ProjectName and respective EmployeeName(firstname) which are working on the project,

--if more then one employee working on same project, then it should be in same cell seprated by comma

--for example :- Task Tracker : Vikas, Ashish

ANS:

```
SELECT ProjectName, STUFF((SELECT  ', ' +  FirstName FROM EmployeeDetail E1 INNER JOIN [ProjectDetail]
P1 ON E1.EmployeeID = P1.EmployeeDetailID
WHERE P1.ProjectName = P2.ProjectName FOR XML PATH('')),1,2,' ) AS [Employee Name] FROM EmployeeDetail E2
INNER JOIN [ProjectDetail] P2 ON E2.EmployeeID = P2.EmployeeDetailID
GROUP BY ProjectName
```

Output:-

	ProjectName	Employee Name
1	CLP	Vikas
2	DDS	Ashish
3	GRS	Ashish
4	HR Managment	nikita, Nikhil
5	Survey Managment	Vikas
6	Task Track	Vikas, Ashish

6). Table Variables vs. Temp Tables?

- Table var doesn't have to be memory-resident. Its pages can be moved to tempdb if memory is low
- Rollback doesn't affect table vars
- Table vars don't participate in transactions or locking
- Any DML operations done on table variables are not logged
- No statistics are maintained on table variables

104) . What would be the out-put of the following Sql query?

```
SELECT A.A FROM (SELECT 1 A, 2 B) A
JOIN (SELECT 1 A,1 B)B ON A.A = B.B
```

Options:

A).
Error

B).

	A

C).

	A
1	1

D).

	A	A	B
1	2	1	2

E).

	A	A	B
1	2	1	2
2	1	1	1

105). What would be the out-put of the following Sql query?

```
SELECT B.A FROM (SELECT 1 A) A
JOIN (SELECT 1 A, 2 B)B ON A.A = B.A
```

Options:

A).
Error

B).

	A

C).

	A
1	1

D).

	A	A	B
1	2	1	2

E).

	A	A	B
1	2	1	2
2	1	1	1

106). What would be the out-put of the following Sql query?

```
SELECT B.A FROM (SELECT 1 A) A
JOIN (SELECT 1 A, 2 B)B ON A.A = B.B
```

Options:

A).
Error

B).

	A

C).

	A
1	1

D).

	A	A	B
1	2	1	2

E).

	A	A	B
1	2	1	2
2	1	1	1

107). What would be the out-put of the following Sql query?

```
SELECT * FROM (SELECT 1 A UNION ALL SELECT 2 B) A
JOIN (SELECT 1 A,2 B UNION ALL SELECT 1 A, 1 B)B ON A.A = B.B
```

Options:

A).
Error

B).

	A

C).

	A
1	1

D).

	A	A	B
1	2	1	2

E).

	A	A	B
1	2	1	2
2	1	1	1

108). What would be the out-put of the following Sql query?

```
SELECT * FROM (SELECT 1 A UNION ALL SELECT 2 B) A
JOIN (SELECT 1 A,2 B)B ON A.A = B.B
```

Options:

A).

Error

B).

	A

C).

	A
1	1

D).

	A	A	B
1	2	1	2

E).

	A	A	B
1	2	1	2
2	1	1	1

Answers: 104) C, 105) C, 106) B, 107) E, 108) D