**Suggested Videos**  
[Part 58 - Transaction Acid Tests](http://csharp-video-tutorials.blogspot.com/2012/10/transaction-acid-tests-part-58.html)  
[Part 59 - Subqueries](http://csharp-video-tutorials.blogspot.com/2013/01/subqueries-in-sql-part-59.html)  
[Part 60 - Correlated subquery](http://csharp-video-tutorials.blogspot.com/2013/01/correlated-subquery-in-sql-part-60.html)   
  
**In this video we will discuss about inserting large amount of random data into sql server tables for performance testing.**  
  
   
  
-- If Table exists drop the tables  
If (Exists (select \*   
            from information\_schema.tables   
            where table\_name = 'tblProductSales'))  
Begin  
 Drop Table tblProductSales  
End  
  
If (Exists (select \*   
            from information\_schema.tables   
            where table\_name = 'tblProducts'))  
Begin  
 Drop Table tblProducts  
End   
  
   
  
-- Recreate tables  
Create Table tblProducts  
(  
 [Id] int identity primary key,  
 [Name] nvarchar(50),  
 [Description] nvarchar(250)  
)  
  
Create Table tblProductSales  
(  
 Id int primary key identity,  
 ProductId int foreign key references tblProducts(Id),  
 UnitPrice int,  
 QuantitySold int  
)  
  
--Insert Sample data into tblProducts table  
Declare @Id int  
Set @Id = 1  
  
While(@Id <= 300000)  
Begin  
 Insert into tblProducts values('Product - ' + CAST(@Id as nvarchar(20)),   
 'Product - ' + CAST(@Id as nvarchar(20)) + ' Description')  
   
 Print @Id  
 Set @Id = @Id + 1  
End  
  
-- Declare variables to hold a random ProductId,   
-- UnitPrice and QuantitySold  
declare @RandomProductId int  
declare @RandomUnitPrice int  
declare @RandomQuantitySold int  
  
-- Declare and set variables to generate a   
-- random ProductId between 1 and 100000  
declare @UpperLimitForProductId int  
declare @LowerLimitForProductId int  
  
set @LowerLimitForProductId = 1  
set @UpperLimitForProductId = 100000  
  
-- Declare and set variables to generate a   
-- random UnitPrice between 1 and 100  
declare @UpperLimitForUnitPrice int  
declare @LowerLimitForUnitPrice int  
  
set @LowerLimitForUnitPrice = 1  
set @UpperLimitForUnitPrice = 100  
  
-- Declare and set variables to generate a   
-- random QuantitySold between 1 and 10  
declare @UpperLimitForQuantitySold int  
declare @LowerLimitForQuantitySold int  
  
set @LowerLimitForQuantitySold = 1  
set @UpperLimitForQuantitySold = 10  
  
--Insert Sample data into tblProductSales table  
Declare @Counter int  
Set @Counter = 1  
  
While(@Counter <= 450000)  
Begin  
 select @RandomProductId = Round(((@UpperLimitForProductId - @LowerLimitForProductId) \* Rand() + @LowerLimitForProductId), 0)  
 select @RandomUnitPrice = Round(((@UpperLimitForUnitPrice - @LowerLimitForUnitPrice) \* Rand() + @LowerLimitForUnitPrice), 0)

select @RandomQuantitySold = Round(((@UpperLimitForQuantitySold - @LowerLimitForQuantitySold) \* Rand() + @LowerLimitForQuantitySold), 0)  
   
 Insert into tblProductsales   
 values(@RandomProductId, @RandomUnitPrice, @RandomQuantitySold)  
  
 Print @Counter  
 Set @Counter = @Counter + 1  
End  
  
**Finally, check the data in the tables using a simple SELECT query** to make sure the data has been inserted as expected.  
Select \* from tblProducts  
Select \* from tblProductSales  
  
**In our next video, we will be using these tables, for performance testing of queries that uses subqueries and joins.**