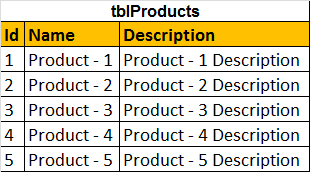
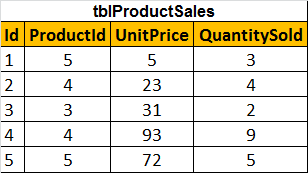
**Suggested Videos**  
[Part 60 - Correlated subquery](http://csharp-video-tutorials.blogspot.com/2013/01/correlated-subquery-in-sql-part-60.html)  
[Part 61 - Creating a large table with random data for performance testing](http://csharp-video-tutorials.blogspot.com/2013/01/creating-large-table-with-random-data.html)  
[Part 62 - What to choose for performance - SubQuery or Joins](http://csharp-video-tutorials.blogspot.com/2013/01/what-to-choose-for-performance.html)   
  
**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**   
  
   
  
**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.  
  
There are different types of cursors in sql server as listed below. We will talk about the differences between these cursor types in a later video session.   
**1.** Forward-Only  
**2.** Static  
**3.** Keyset  
**4.** Dynamic    
  
   
  
**Let us now look at a simple example of using sql server cursor to process one row at time.** We will be using tblProducts and tblProductSales tables, for this example. The tables here show only 5 rows from each table. However, on my machine, there are 400,000 records in tblProducts and 600,000 records in tblProductSales tables. If you want to learn about generating huge amounts of random test data, [**please watch Part - 61 in sql server video tutorial.**](http://www.youtube.com/user/kudvenkat/videos?flow=grid&view=1)   
  
   
  
   
  
**Cursor Example:** Let us say, I want to update the UNITPRICE column in tblProductSales table, based on the following criteria  
**1.** If the ProductName = 'Product - 55', Set Unit Price to 55  
**2.** If the ProductName = 'Product - 65', Set Unit Price to 65  
**3.** If the ProductName is like 'Product - 100%', Set Unit Price to 1000  
  
Declare @ProductId int  
  
-- Declare the cursor using the declare keyword  
Declare ProductIdCursor CURSOR FOR   
Select ProductId from tblProductSales  
  
-- Open statement, executes the SELECT statment  
-- and populates the result set  
Open ProductIdCursor  
  
-- Fetch the row from the result set into the variable  
Fetch Next from ProductIdCursor into @ProductId  
  
-- If the result set still has rows, @@FETCH\_STATUS will be ZERO  
While(@@FETCH\_STATUS = 0)  
Begin  
 Declare @ProductName nvarchar(50)  
 Select @ProductName = Name from tblProducts where Id = @ProductId  
   
 if(@ProductName = 'Product - 55')  
 Begin  
 Update tblProductSales set UnitPrice = 55 where ProductId = @ProductId  
 End  
 else if(@ProductName = 'Product - 65')  
 Begin  
 Update tblProductSales set UnitPrice = 65 where ProductId = @ProductId  
 End  
 else if(@ProductName like 'Product - 100%')  
 Begin  
 Update tblProductSales set UnitPrice = 1000 where ProductId = @ProductId  
 End  
   
 Fetch Next from ProductIdCursor into @ProductId   
End  
  
-- Release the row set  
CLOSE ProductIdCursor   
-- Deallocate, the resources associated with the cursor  
DEALLOCATE ProductIdCursor  
  
The cursor will loop thru each row in tblProductSales table. As there are 600,000 rows, to be processed on a row-by-row basis, it takes around 40 to 45 seconds on my machine. We can achieve this very easily using a join, and this will significantly increase the performance. We will discuss about this in our next video session.  
  
**To check if the rows have been correctly updated, please use the following query.**  
Select  Name, UnitPrice   
from tblProducts join  
tblProductSales on tblProducts.Id = tblProductSales.ProductId  
where (Name='Product - 55' or Name='Product - 65' or Name like 'Product - 100%')