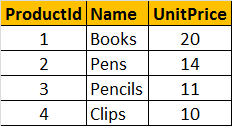
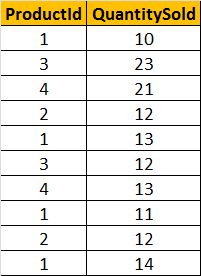
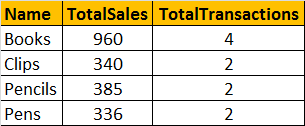
**Suggested SQL Server Videos before watching this Video**  
1. [Part 39 - Views in sql server](http://csharp-video-tutorials.blogspot.com/2012/09/views-in-sql-server-part-39.html)  
2. [Part 40 - Updateable views in sql server](http://csharp-video-tutorials.blogspot.com/2012/09/updateable-views-part-40.html)   
  
   
  
   
  
   
  
In [**Part 39**](http://csharp-video-tutorials.blogspot.com/2012/09/views-in-sql-server-part-39.html), we have covered the basics of views and in [**Part 40**](http://csharp-video-tutorials.blogspot.com/2012/09/updateable-views-part-40.html), we have seen, how to update the underlying base tables thru a view. In this video session, we will learn about 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.  
  
**Let's now, look at an example of creating an Indexed view**. For the purpose of this video, we will be using **tblProduct** and **tblProductSales** tables.  
  
**Script to create table tblProduct**  
Create Table tblProduct  
(  
 ProductId int primary key,  
 Name nvarchar(20),  
 UnitPrice int  
)  
  
**Script to pouplate tblProduct, with sample data**  
Insert into tblProduct Values(1, 'Books', 20)  
Insert into tblProduct Values(2, 'Pens', 14)  
Insert into tblProduct Values(3, 'Pencils', 11)  
Insert into tblProduct Values(4, 'Clips', 10)  
  
**Script to create table tblProductSales**  
Create Table tblProductSales  
(  
 ProductId int,  
 QuantitySold int  
)  
  
**Script to pouplate tblProductSales, with sample data**  
Insert into tblProductSales values(1, 10)  
Insert into tblProductSales values(3, 23)  
Insert into tblProductSales values(4, 21)  
Insert into tblProductSales values(2, 12)  
Insert into tblProductSales values(1, 13)  
Insert into tblProductSales values(3, 12)  
Insert into tblProductSales values(4, 13)  
Insert into tblProductSales values(1, 11)  
Insert into tblProductSales values(2, 12)  
Insert into tblProductSales values(1, 14)   
  
**tblProduct Table**  
   
  
**tblProductSales Table**  
  
  
**Create a view which returns Total Sales and Total Transactions by Product.** The output should be, as shown below.   
   
  
**Script to create view vWTotalSalesByProduct**  
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  
  
**If you want to create an Index**, on a view, the following rules should be followed by the view. For the complete list of all rules, please check [MSDN](http://msdn.microsoft.com/en-us/library/ms191432(v=sql.105).aspx).  
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. In this example, we are using, ISNULL() function, to replace NULL values with ZERO.  
  
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. In this example, tblProduct and tblProductSales are referenced using dbo.tblProduct and dbo.tblProductSales respectively.  
  
**Now, let's create an Index on the view:**  
The first index that you create on a view, must be a unique clustered index. After the unique clustered index has been created, you can create additional nonclustered indexes.  
Create Unique Clustered Index UIX\_vWTotalSalesByProduct\_Name  
on vWTotalSalesByProduct(Name)   
  
**Since, we now have an index on the view, the view gets materialized**. The data is stored in the view. So when we execute Select \* from vWTotalSalesByProduct, the data is retrurned from the view itself, rather than retrieving data from the underlying base tables.   
  
Indexed views, can significantly improve the performance of queries that involves JOINS and Aggeregations. The cost of maintaining an indexed view is much higher than the cost of maintaining a table index.   
  
Indexed views are ideal for scenarios, where the underlying data is not frequently changed. Indexed views are more often used in OLAP systems, because the data is mainly used for reporting and analysis purposes. Indexed views, may not be suitable for OLTP systems, as the data is frequently addedd and changed.