**Suggested Videos**  
[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)  
[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)   
  
According to MSDN, in sql server, in most cases, there is usually no performance difference between queries that uses sub-queries and equivalent queries using joins. For example, on my machine I have  
**400,000 records in tblProducts table**  
**600,000 records in tblProductSales tables**   
  
   
  
**The following query, returns, the list of products that we have sold atleast once.**This query is formed using sub-queries. When I execute this query I get 306,199 rows in 6 seconds  
Select Id, Name, Description  
from tblProducts  
where ID IN  
(  
 Select ProductId from tblProductSales  
)   
  
   
  
**At this stage please clean the query and execution plan cache using the following T-SQL command.**  
CHECKPOINT;   
GO   
DBCC DROPCLEANBUFFERS; -- Clears query cache  
Go  
DBCC FREEPROCCACHE; -- Clears execution plan cache  
GO  
  
**Now, run the query that is formed using joins.** Notice that I get the exact same 306,199 rows in 6 seconds.   
  
Select distinct tblProducts.Id, Name, Description  
from tblProducts  
inner join tblProductSales  
on tblProducts.Id = tblProductSales.ProductId  
  
**Please Note:** I have used automated sql script to insert huge amounts of this random data. Please watch Part 61 of SQL Server tutorial, in which we have discussed about this automated script.  
  
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.  
  
The following query returns the products that we have not sold at least once. This query is formed using sub-queries. When I execute this query I get 93,801 rows in 3 seconds  
  
Select Id, Name, [Description]  
from tblProducts  
where Not Exists(Select \* from tblProductSales where ProductId = tblProducts.Id)  
  
**When I execute the below equivalent query**, that uses joins, I get the exact same 93,801 rows in 3 seconds.  
  
Select tblProducts.Id, Name, [Description]  
from tblProducts  
left join tblProductSales  
on tblProducts.Id = tblProductSales.ProductId  
where tblProductSales.ProductId IS NULL   
  
In general joins work faster than sub-queries, but in reality it all depends on the execution plan that is generated by SQL Server. It does not matter how we have written the query, SQL Server will always transform it on an execution plan. If sql server generates the same plan from both queries, we 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.