```
----Note: the following code clears the previous execution plan from the system and should be used for testing
CHECKPOINT;
GO
DBCC DROPCLEANBUFFERS; ---Clears query cache
GO
DBCC FREEPROCCACHE; ---Clears execution plan cache
GO

-----if the tables tblProducts and tblProductSales exist then delete them

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;

------ create tables tblProducts and tblPRoductSales

create table tblProducts
(
   Id int identity primary key,
   Name varchar(50),
   [Description] varchar(100)
)

create table tblProductSales
(
    id int primary key identity,
       ProductId int foreign key references tblProducts(Id),
       UnitPrice money,
       QuantitySold int
)


------------------------------------------------------------
```

```
-------- Inserting five hundred thousand rows for testing
Declare @Id int
Set @Id = 1
While(@Id <= 500000)
Begin
   Insert into tblProducts values('ProductName-' + CAST(@Id as varchar(20)),
   'ProductDesc-' + CAST(@Id as varchar(20)))
   Print @Id
   Set @Id = @Id + 1
End
select * from tblProducts
----------------------------------------------------------------------
---declare variables to hold a random productId, unit price qtysold
declare @RandomProductId int
declare @RandomUnitPrice money
declare @RandomQuantitySold int
----declare and set variables to generate a random prodid between 1 and 300000 leaving some items unsold in tblProducts
declare @UpperLimitForProdId int
declare @LowerLimitForProdId int
set @LowerLimitForProdId = 1
set @UpperLimitForProdId = 300000
--------
declare @UpperLimitForQtySold int
declare @LowerLimitForQtySold int
set @LowerLimitForQtySold = 1
set @UpperLimitForQtySold = 10
-------- set upper unit price as $100.00
declare @UpperLimitUnitPrice int
declare @LowerLimitUnitPrice int
set @LowerLimitUnitPrice = 1
set @UpperLimitUnitPrice =  100
---- insert one million sales into tblProductSales
declare @Counter int
set @Counter = 1
While(@Counter <= 500000)
Begin

       select @RandomProductId = Round(((@UpperLimitForProdId - @LowerLimitForProdId) * RAND() + @LowerLimitForProdId),0)
       select @RandomQuantitySold =   Round(((@UpperLimitForQtySold - @LowerLimitForQtySold) * RAND() +
@LowerLimitForQtySold),0)
       select @RandomUnitPrice =  Round(((@UpperLimitUnitPrice - @LowerLimitUnitPrice) * RAND() + @LowerLimitUnitPrice),0)
       insert into tblProductSales values (@RandomProductId,@RandomUnitPrice,@RandomQuantitySold)
```

```
       print @Counter
       set @Counter = @Counter + 1
End

select * from tblProductSales
------ Which is better a subquery or a join?


----Note: the following code clears the previous execution plan from the system and should be used for testing
CHECKPOINT;
GO
DBCC DROPCLEANBUFFERS; ---Clears query cache
GO
DBCC FREEPROCCACHE; ---Clears execution plan cache
GO


--- USE 'CLIENT STATISTICS' AND 'ACTUAL EXECUTION PLAN' TO COMPARE METHODS
---- subquery
---3 seconds returning 243276 records for test 1
select
Id,
Name
from
tblProducts
where Id in (select productId from tblProductSales);

---- join
select
distinct
tblProducts.Id,
Name
from
tblProducts
inner join tblProductSales
on tblProducts.id = tblProductSales.ProductId;

CHECKPOINT;
GO
DBCC DROPCLEANBUFFERS; ---Clears query cache
GO
DBCC FREEPROCCACHE; ---Clears execution plan cache
```

```
GO

----- subquery vs. join testing
---subquery
select
Id,
Name,
[Description]
from
tblProducts
where Not Exists(select * from tblProductSales where ProductId = tblProducts.Id);

---Join
Select
tblProducts.Id,
Name,
[Description]
from
tblProducts
left join tblProductSales
on tblProducts.Id = tblProductSales.ProductId
where tblProductSales.ProductId IS NULL;
```