(T20)大量 Data 的 Performance。比較 Sub-Query 和 Join
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc
==
(T20)大量 Data 的 Performance。比較 Sub-Query 和 Join
==

==

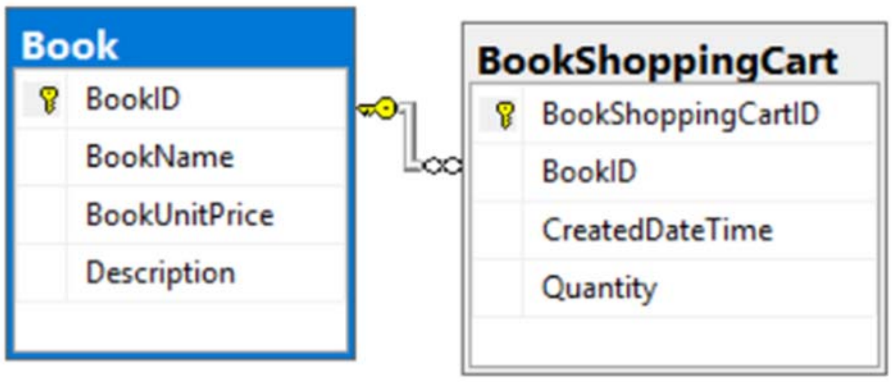
0. Summary

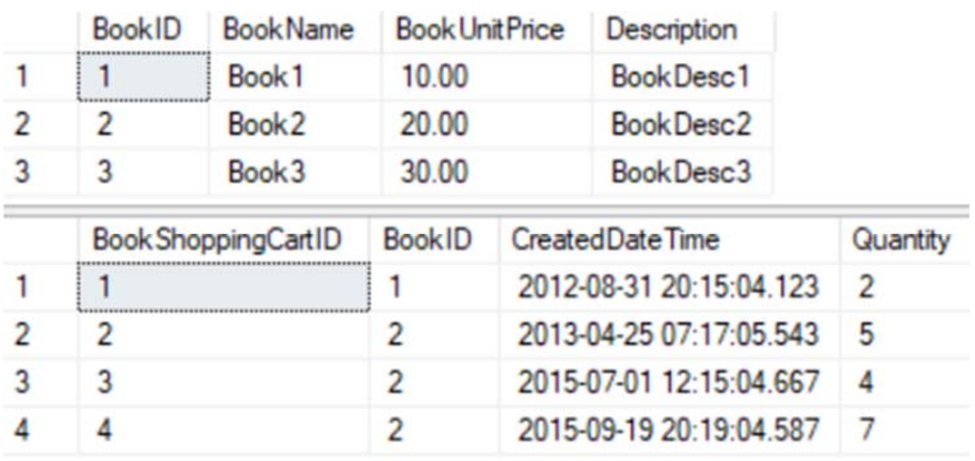
1.
```
CHECKPOINT;
GO
-- Clears query cache
DBCC DROPCLEANBUFFERS;
GO
-- Clears execution plan cache
DBCC FREEPROCCACHE;
GO
```
2.
Random Number
2.1.
RAND([seed])
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/functions/rand-transact-sql
https://www.w3schools.com/sql/func_mysql_rand.asp
Returns a pseudo-random float value from 0 through 1, exclusive.
0 <= ReturnNumber < 1
Same seed always returns the same RAND([seed]) value.
2.2.
FLOOR(RAND()*(b-a)+a);
Where a is the smallest number and b is the largest number that you want to generate a random number for.
Reference:
https://www.techonthenet.com/sql_server/functions/rand.php
```
PRINT FLOOR(RAND()*(25-10)+10);
```

10 <= IntNumber < 25
3.
Random DateTime
--Ch25_08
--Get Random DateTime
--Reference: http://crodrigues.com/sql-server-generate-random-datetime-within-a-range/
DECLARE @RandomDateTime DATETIME;
DECLARE @DateFrom DATETime = '2012-01-01'
DECLARE @DateTo DATeTime = '2017-06-30'
DECLARE @DaysRandom Int= 0
DECLARE @MillisRandom Int=0
--get random number of days
select @DaysRandom= DATEDIFF(day,@DateFrom,@DateTo)
SELECT @DaysRandom = ROUND(((@DaysRandom -1) * RAND()), 0)
--get random millis
SELECT @MillisRandom = ROUND(((99999999) * RAND()), 0)
SELECT @RandomDateTime = DATEADD(day, @DaysRandom, @DateFrom)
SELECT @RandomDateTime = DATEADD(MILLISECOND, @MillisRandom, @RandomDateTime)
SELECT @RandomDateTime
4.
Theoretically, joins is faster than sub-queries.
In reality, SQL Server always transforms query on an execution plan.
If sql server generates the same execution plan from both queries,
then it will return the same result.
It is always better to do real testing and make a decision.

===

1. Create Sample Data





	BookID	BookName	BookUnitPrice	Description
1	1	Book1	10.00	BookDesc1
2	2	Book2	20.00	BookDesc2
3	3	Book3	30.00	BookDesc3

	BookShoppingCartID	BookID	CreatedDateTime	Quantity
1	1	1	2012-08-31 20:15:04.123	2
2	2	2	2013-04-25 07:17:05.543	5
3	3	2	2015-07-01 12:15:04.667	4
4	4	2	2015-09-19 20:19:04.587	7

```
--========================================================================
--T020_01_Create Sample Data
--========================================================================
```

```sql
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'BookShoppingCart' ) )
   BEGIN
       TRUNCATE TABLE BookShoppingCart;
       DROP TABLE BookShoppingCart;
   END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Book' ) )
   BEGIN
       TRUNCATE TABLE Book;
       DROP TABLE Book;
   END;
GO -- Run the previous command and begins new batch
CREATE TABLE Book
(
  BookID INT PRIMARY KEY
              IDENTITY(1, 1)
              NOT NULL ,
  BookName NVARCHAR(100) NULL ,
  BookUnitPrice MONEY NULL ,
  [Description] NVARCHAR(1000) NULL,
 );
GO -- Run the previous command and begins new batch
INSERT  INTO Book
VALUES  ( 'Book1', 10, 'BookDesc1' );
INSERT  INTO Book
VALUES  ( 'Book2', 20, 'BookDesc2' );
INSERT  INTO Book
VALUES  ( 'Book3', 30, 'BookDesc3' );
GO -- Run the previous command and begins new batch
CREATE TABLE BookShoppingCart
   (
      BookShoppingCartID INT PRIMARY KEY
                             IDENTITY(1, 1)
                             NOT NULL ,
      BookID INT FOREIGN KEY REFERENCES Book ( [BookID] )
                 NOT NULL ,
      CreatedDateTime DATETIME NULL ,
      Quantity INT NULL,
   )
GO -- Run the previous command and begins new batch
INSERT  INTO BookShoppingCart
VALUES  ( 1, '2012-08-31 20:15:04.123', 2 );
INSERT  INTO BookShoppingCart
VALUES  ( 2, '2013-04-25 07:17:05.543', 5 );
INSERT  INTO BookShoppingCart
VALUES  ( 2, '2015-07-01 12:15:04.667', 4 );
INSERT  INTO BookShoppingCart
VALUES  ( 2, '2015-09-19 20:19:04.588', 7 );
GO -- Run the previous command and begins new batch
```

```
SELECT  *
FROM    Book;
SELECT  *
FROM    BookShoppingCart;
GO -- Run the previous command and begins new batch
```

==

2. Get the book that has never been sold

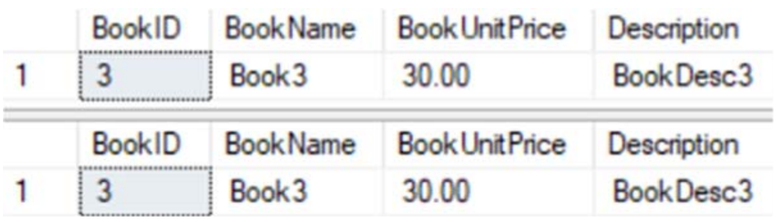
```
--===========================================================================
--T020_02_Get the book that has never been sold
--===========================================================================
```

2.1. GET the book that has never been sold - SubQuery

```
--===========================================================================
--T020_02_01
--GET the book that has never been sold - SubQuery
SELECT  b.BookID ,
        b.BookName ,
        b.BookUnitPrice ,
        b.[Description]
FROM    Book b
WHERE   b.BookID NOT IN ( SELECT DISTINCT
                                    bsc.BookID
                          FROM      BookShoppingCart bsc );
GO -- Run the previous command and begins new batch
```

2.2. Get the book that has never been sold - JOIN

```
--===========================================================================
--T020_02_02
--Get the book that has never been sold - JOIN
SELECT  b.BookID ,
        b.BookName ,
        b.BookUnitPrice ,
        b.[Description]
FROM    Book b
        LEFT JOIN BookShoppingCart bsc ON b.BookID = bsc.BookID
WHERE   bsc.BookID IS NULL;
GO -- Run the previous command and begins new batch
/*
Reference:
https://technet.microsoft.com/en-us/library/ms189575(v=sql.105).aspx
subqueries can be nested upto 32 levels.
*/
```



	BookID	BookName	BookUnitPrice	Description
1	3	Book3	30.00	BookDesc3

	BookID	BookName	BookUnitPrice	Description
1	3	Book3	30.00	BookDesc3

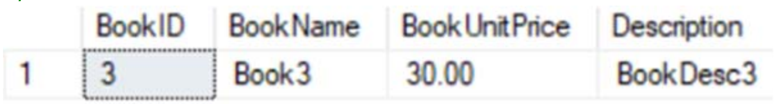
==

3. CorrelatedSubquery V.S. NonCorrelatedSubquery

```
--==========================================================================
--T020_03_CorrelatedSubquery V.S. NonCorrelatedSubquery
--==========================================================================
```

3.1. non-corelated sub-query

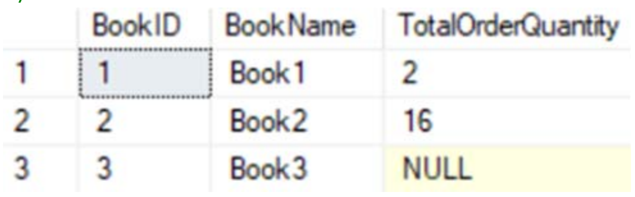
```
--==========================================================================
--T020_03_01
--non-corelated sub-query
SELECT  b.BookID ,
        b.BookName ,
        b.BookUnitPrice ,
        b.[Description]
FROM    Book b
WHERE   b.BookID NOT IN ( SELECT DISTINCT
                                    bsc.BookID
                          FROM      BookShoppingCart bsc );
GO -- Run the previous command and begins new batch
/*
A non-corelated sub-query can be executed independently.
E.g.
--SELECT DISTINCT bsc.BookID
--FROM   BookShoppingCart bsc
*/
```



	BookID	BookName	BookUnitPrice	Description
1	3	Book3	30.00	BookDesc3

3.2. corelated sub-query

```
--==========================================================================
--T020_03_02
--corelated sub-query
SELECT  b.BookID ,
        b.BookName ,
        ( SELECT    SUM(bsc.Quantity)
          FROM      BookShoppingCart bsc
          WHERE     b.BookID = bsc.BookID
        ) AS TotalOrderQuantity
FROM    Book b
ORDER BY b.BookName;
GO -- Run the previous command and begins new batch
/*
A corelated sub-query can NOT be executed independently,
because sub-query depends on the value of outer query.
E.g.
--SELECT    SUM(bsc.Quantity)
--FROM      BookShoppingCart bsc
--WHERE     b.BookID = bsc.BookID
*/
```



	BookID	BookName	TotalOrderQuantity
1	1	Book1	2
2	2	Book2	16
3	3	Book3	NULL

==

4. PerformanceTesting

```
--==========================================================================
--T020_04_PerformanceTesting
--==========================================================================
```

4.1. Create large amount of data

```
--==========================================================================
--T020_04_01
--Create large amount of data
--T020_04_01_01
--Create Table
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'BookShoppingCart' ) )
    BEGIN
        DROP TABLE BookShoppingCart;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Book' ) )
    BEGIN
        DROP TABLE Book;
    END;
GO -- Run the previous command and begins new batch
CREATE TABLE Book
(
  BookID INT PRIMARY KEY
             IDENTITY(1, 1)
             NOT NULL ,
  BookName NVARCHAR(100) NULL ,
  BookUnitPrice MONEY NULL ,
  [Description] NVARCHAR(1000) NULL,
 );
GO -- Run the previous command and begins new batch
CREATE TABLE BookShoppingCart
(
  BookShoppingCartID INT PRIMARY KEY
                         IDENTITY(1, 1)
                         NOT NULL ,
  BookID INT FOREIGN KEY REFERENCES Book ( [BookID] )
             NOT NULL ,
  CreatedDateTime DATETIME NULL ,
  Quantity INT NULL,
 );
GO -- Run the previous command and begins new batch
--------------------------------------------
--T020_04_01_02
--Insert to Book
--Whole T020_04_01 part need to execute together.
--Book Counter
DECLARE @TotalBookRows INT = 300000;
DECLARE @BookCount INT = 1;
```

```
-- random UnitPrice between 1 and 100
DECLARE @RandomUnitPrice MONEY;
DECLARE @BookUnitPrice_Max INT;
DECLARE @BookUnitPrice_Min INT;
SET @BookUnitPrice_Min = 1;
SET @BookUnitPrice_Max = 100;
WHILE ( @BookCount <= @TotalBookRows )
    BEGIN
        SELECT  @RandomUnitPrice = FLOOR(RAND() * ( @BookUnitPrice_Max
                                                    - @BookUnitPrice_Min )
                                             + @BookUnitPrice_Min);
        INSERT  INTO Book
        VALUES  ( 'Book ' + CAST(@BookCount AS NVARCHAR(20)),
                  @RandomUnitPrice,
                  'Book Description ' + CAST(@BookCount AS NVARCHAR(20)) );
        PRINT @BookCount;
        SET @BookCount += 1;
    END;
/*
1.
Random Number
1.1.
RAND([seed])
Reference:
https://docs.microsoft.com/en-us/sql/t-sql/functions/rand-transact-sql
https://www.w3schools.com/sql/func_mysql_rand.asp
Returns a pseudo-random float value from 0 through 1, exclusive.
0 <= ReturnNumber < 1
Same seed always returns the same RAND([seed]) value.
1.2.
FLOOR(RAND()*(b-a)+a);
Where a is the smallest number and b is the largest number that you want to generate a random number for.
Reference:
https://www.techonthenet.com/sql_server/functions/rand.php
PRINT FLOOR(RAND()*(25-10)+10);
10 <= IntNumber < 25
2.
Random DateTime
--Ch25_08
--Get Random DateTime
--Reference: http://crodrigues.com/sql-server-generate-random-datetime-within-a-range/
DECLARE @RandomDateTime DATETIME;
DECLARE @DateFrom DATETime = '2012-01-01'
DECLARE @DateTo DATeTime = '2017-06-30'
DECLARE @DaysRandom Int= 0
DECLARE @MillisRandom Int=0
--get random number of days
select @DaysRandom= DATEDIFF(day,@DateFrom,@DateTo)
SELECT @DaysRandom = ROUND(((@DaysRandom -1) * RAND()), 0)
--get random millis
SELECT @MillisRandom = ROUND(((99999999) * RAND()), 0)
SELECT @RandomDateTime = DATEADD(day, @DaysRandom, @DateFrom)
SELECT @RandomDateTime = DATEADD(MILLISECOND, @MillisRandom, @RandomDateTime)
SELECT @RandomDateTime
*/
--------------------------------------------
--T020_04_01_02
--Insert sample data to [BookShoppingCart] table
--Whole T020_04_01 part need to execute together.
--BookShoppingCart Counter
DECLARE @TotalBookShoppingCartRows INT;
DECLARE @BookShoppingCartCount INT;
```

```sql
SET @BookShoppingCartCount = 1;
SET @TotalBookShoppingCartRows = 400000;
-- @RandomBookID
DECLARE @RandomBookID INT;
DECLARE @RandomBookID_Max INT;
DECLARE @RandomBookID_Min INT;
SET @RandomBookID_Min = 1;
SET @RandomBookID_Max = @TotalBookRows - 100;
--Should be @RandomBookID_Max = @TotalBookRows,
--but I purposely set  @RandomBookID_Max = @TotalBookRows-100
--I want some book data that was never sold.
--@RandomCreatedDateTime
--Reference: http://crodrigues.com/sql-server-generate-random-datetime-within-a-range/
DECLARE @RandomCreatedDateTime DATETIME;
DECLARE @DateFrom DATETIME = '2012-01-01';
DECLARE @DateTo DATETIME = '2017-06-30';
DECLARE @DaysRandom INT= 0;
DECLARE @MillisRandom INT= 0;
-- @RandomQuantity is between 1 to 10
DECLARE @RandomQuantity INT;
DECLARE @RandomQuantity_Max INT;
DECLARE @RandomQuantity_Min INT;
SET @RandomQuantity_Min = 1;
SET @RandomQuantity_Max = 10;
WHILE ( @BookShoppingCartCount <= @TotalBookShoppingCartRows )
    BEGIN
            --1. @RandomBookID
        SELECT  @RandomBookID = FLOOR(RAND() * ( @RandomBookID_Max
                                              - @RandomBookID_Min )
                                   + @RandomBookID_Min);
            --2. @RandomQuantity
        SELECT  @RandomQuantity = FLOOR(RAND() * ( @RandomQuantity_Max
                                                - @RandomQuantity_Min )
                                     + @RandomQuantity_Min);
            --3. @RandomCreatedDateTime
            --get random number of days
        SELECT  @DaysRandom = DATEDIFF(DAY, @DateFrom, @DateTo);
        SELECT  @DaysRandom = ROUND(( ( @DaysRandom - 1 ) * RAND() ), 0);
            --get random millis
        SELECT  @MillisRandom = ROUND(( ( 99999999 ) * RAND() ), 0);
        SELECT  @RandomCreatedDateTime = DATEADD(DAY, @DaysRandom, @DateFrom);
        SELECT  @RandomCreatedDateTime = DATEADD(MILLISECOND, @MillisRandom,
                                              @RandomCreatedDateTime);
        INSERT  INTO BookShoppingCart
        VALUES  ( @RandomBookID, @RandomCreatedDateTime, @RandomQuantity );
        PRINT @BookShoppingCartCount;
        SET @BookShoppingCartCount += 1;
    END;
GO -- Run the previous command and begins new batch
```

4.2. Select ...

```sql
--==================================================================
--T020_04_02
SELECT  *
FROM    Book;
SELECT  *
```

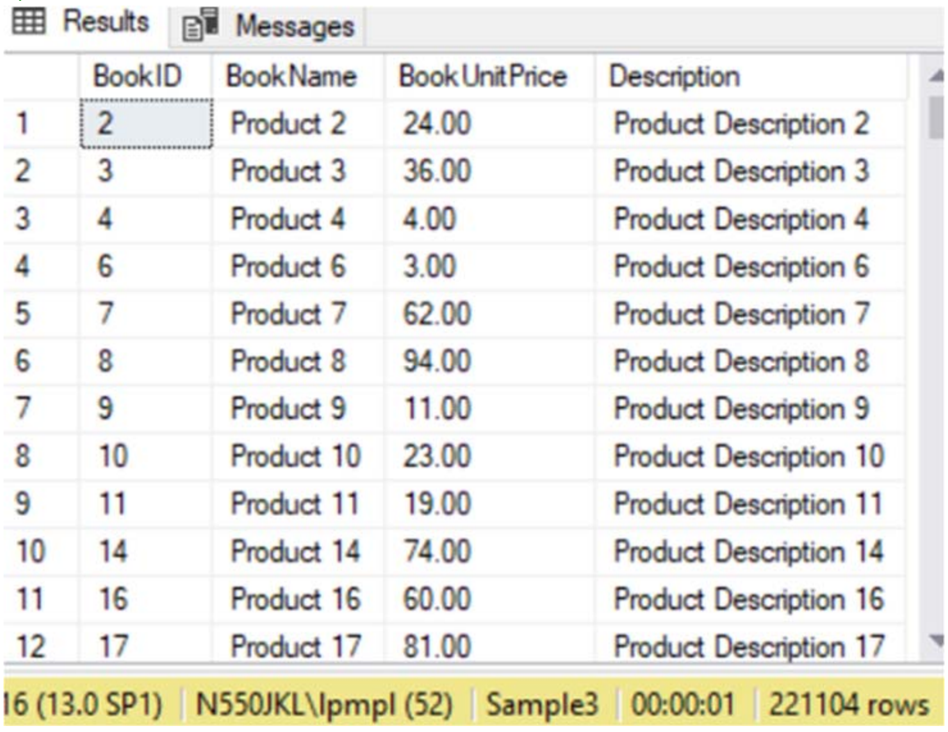
```
FROM    BookShoppingCart;
GO -- Run the previous command and begins new batch

=====================================================
```

5. SubQuery V.S. JoinsPerformance

```
--==========================================================================
--T020_05_SubQuery V.S. JoinsPerformance
--==========================================================================
```

5.1. Compare Join V.S. SubQuery

```
--==========================================================================
--T020_05_01
--Compare Join V.S. SubQuery
SELECT  b.BookID ,
        b.BookName ,
        b.BookUnitPrice ,
        b.[Description]
FROM    Book b
WHERE   b.BookID IN ( SELECT    bsc.BookID
                      FROM      BookShoppingCart bsc );
GO -- Run the previous command and begins new batch
/*
Run 221073 rows in 1 second.
*/
```



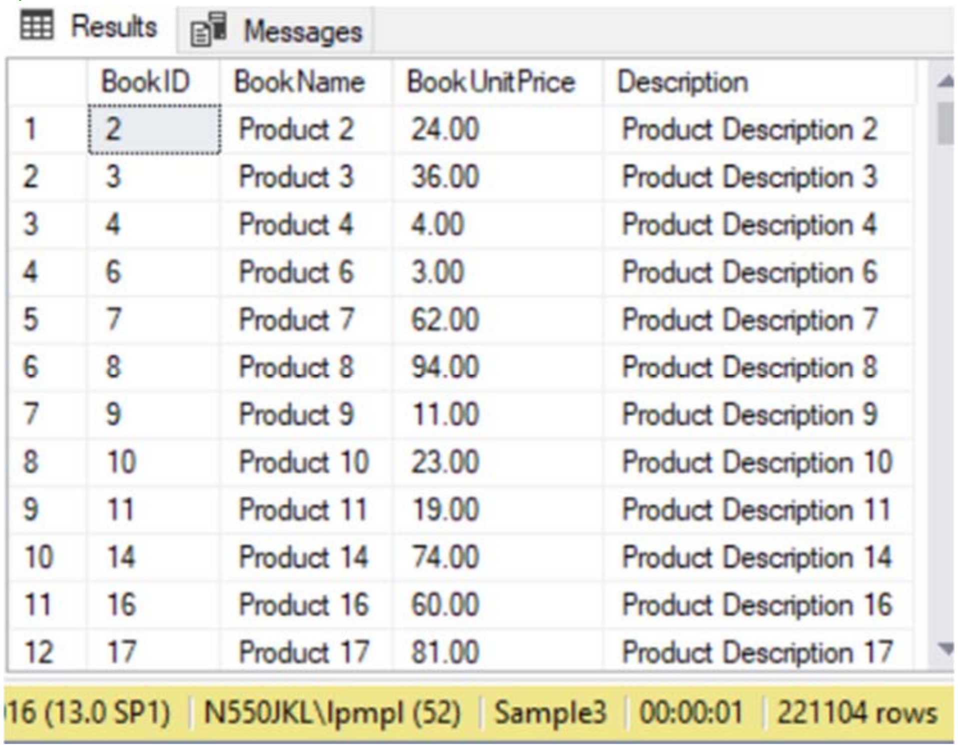
Results | Messages

	BookID	BookName	BookUnitPrice	Description
1	2	Product 2	24.00	Product Description 2
2	3	Product 3	36.00	Product Description 3
3	4	Product 4	4.00	Product Description 4
4	6	Product 6	3.00	Product Description 6
5	7	Product 7	62.00	Product Description 7
6	8	Product 8	94.00	Product Description 8
7	9	Product 9	11.00	Product Description 9
8	10	Product 10	23.00	Product Description 10
9	11	Product 11	19.00	Product Description 11
10	14	Product 14	74.00	Product Description 14
11	16	Product 16	60.00	Product Description 16
12	17	Product 17	81.00	Product Description 17

16 (13.0 SP1) | N550JKL\lpmpl (52) | Sample3 | 00:00:01 | 221104 rows

```
CHECKPOINT;
GO -- Run the previous command and begins new batch
-- Clears query cache
DBCC DROPCLEANBUFFERS;
GO -- Run the previous command and begins new batch
-- Clears execution plan cache
DBCC FREEPROCCACHE;
GO -- Run the previous command and begins new batch
SELECT DISTINCT
```

```
        b.BookID ,
        b.BookName ,
        b.BookUnitPrice ,
        b.[Description]
FROM    Book b
        INNER JOIN BookShoppingCart bsc ON b.BookID = bsc.BookID;
GO -- Run the previous command and begins new batch
/*
Run 221073 rows in 1 second.
*/
```



Results | Messages

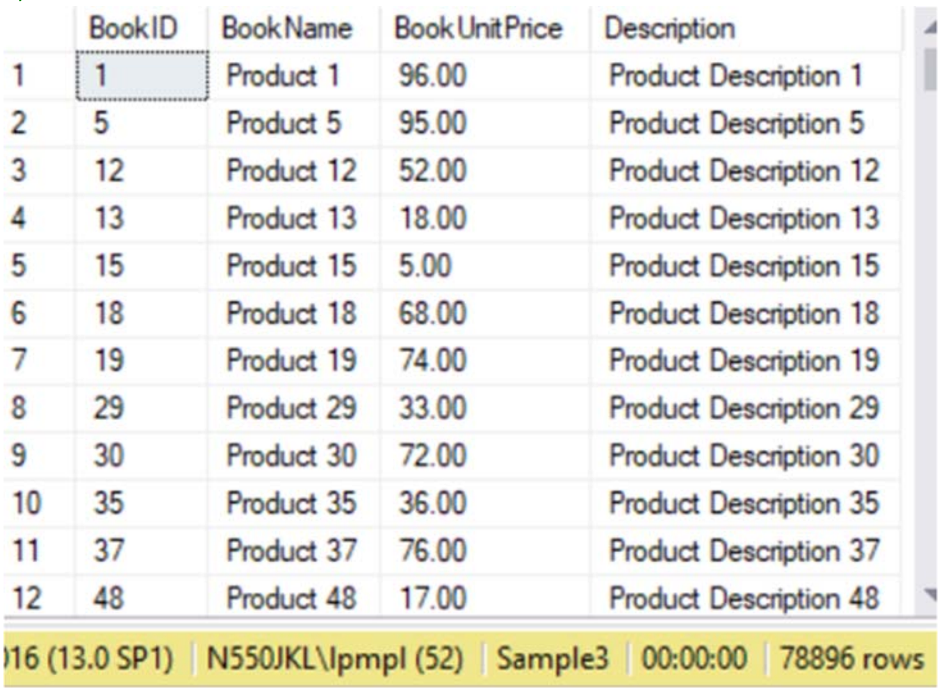
	BookID	BookName	BookUnitPrice	Description
1	2	Product 2	24.00	Product Description 2
2	3	Product 3	36.00	Product Description 3
3	4	Product 4	4.00	Product Description 4
4	6	Product 6	3.00	Product Description 6
5	7	Product 7	62.00	Product Description 7
6	8	Product 8	94.00	Product Description 8
7	9	Product 9	11.00	Product Description 9
8	10	Product 10	23.00	Product Description 10
9	11	Product 11	19.00	Product Description 11
10	14	Product 14	74.00	Product Description 14
11	16	Product 16	60.00	Product Description 16
12	17	Product 17	81.00	Product Description 17

16 (13.0 SP1) | N550JKL\lpmpl (52) | Sample3 | 00:00:01 | 221104 rows

```
CHECKPOINT;
GO -- Run the previous command and begins new batch
-- Clears query cache
DBCC DROPCLEANBUFFERS;
GO -- Run the previous command and begins new batch
-- Clears execution plan cache
DBCC FREEPROCCACHE;
GO -- Run the previous command and begins new batch
```

5.2. Compare Join V.S. SubQuery

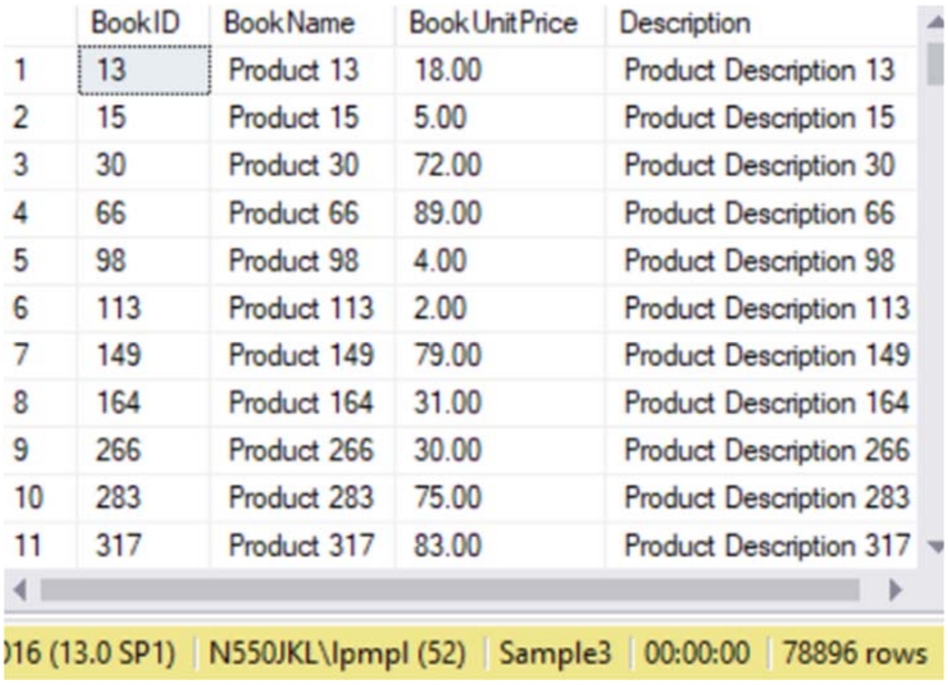
```
--===========================================================================
--T020_05_02
--Compare Join V.S. SubQuery
/*
Theoretically, joins is faster than sub-queries.
In reality, SQL Server always transforms query on an execution plan.
If sql server generates the same execution plan from both queries,
then it will return the same result.
It is alwys better to do real testing and make a decision.
*/
SELECT  b.BookID ,
        b.BookName ,
        b.BookUnitPrice ,
        b.[Description]
FROM    Book b
WHERE   b.BookID NOT IN ( SELECT    bsc.BookID
                    FROM       BookShoppingCart bsc );
GO -- Run the previous command and begins new batch
/*
Run 78927 rows less than 1 second.
*/
```



	BookID	BookName	BookUnitPrice	Description
1	1	Product 1	96.00	Product Description 1
2	5	Product 5	95.00	Product Description 5
3	12	Product 12	52.00	Product Description 12
4	13	Product 13	18.00	Product Description 13
5	15	Product 15	5.00	Product Description 15
6	18	Product 18	68.00	Product Description 18
7	19	Product 19	74.00	Product Description 19
8	29	Product 29	33.00	Product Description 29
9	30	Product 30	72.00	Product Description 30
10	35	Product 35	36.00	Product Description 35
11	37	Product 37	76.00	Product Description 37
12	48	Product 48	17.00	Product Description 48

)16 (13.0 SP1) | N550JKL\lpmpl (52) | Sample3 | 00:00:00 | 78896 rows

```
CHECKPOINT;
GO -- Run the previous command and begins new batch
-- Clears query cache
DBCC DROPCLEANBUFFERS;
GO -- Run the previous command and begins new batch
-- Clears execution plan cache
DBCC FREEPROCCACHE;
GO -- Run the previous command and begins new batch
SELECT DISTINCT
        b.BookID ,
        b.BookName ,
        b.BookUnitPrice ,
        b.[Description]
FROM    Book b
        LEFT JOIN BookShoppingCart bsc ON b.BookID = bsc.BookID
WHERE   bsc.BookID IS NULL;
GO -- Run the previous command and begins new batch
/*
Run 78927 rows less than 1 second.
*/
```



	BookID	BookName	BookUnitPrice	Description
1	13	Product 13	18.00	Product Description 13
2	15	Product 15	5.00	Product Description 15
3	30	Product 30	72.00	Product Description 30
4	66	Product 66	89.00	Product Description 66
5	98	Product 98	4.00	Product Description 98
6	113	Product 113	2.00	Product Description 113
7	149	Product 149	79.00	Product Description 149
8	164	Product 164	31.00	Product Description 164
9	266	Product 266	30.00	Product Description 266
10	283	Product 283	75.00	Product Description 283
11	317	Product 317	83.00	Product Description 317

```
)16 (13.0 SP1) | N550JKL\lpmpl (52) | Sample3 | 00:00:00 | 78896 rows
```

```
CHECKPOINT;
GO -- Run the previous command and begins new batch
-- Clears query cache
DBCC DROPCLEANBUFFERS;
GO -- Run the previous command and begins new batch
-- Clears execution plan cache
DBCC FREEPROCCACHE;
GO -- Run the previous command and begins new batch
```

===

6. Clean up

```
--=========================================================================
--T020_06_Clean up
--=========================================================================
--Clean up
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'BookShoppingCart' ) )
    BEGIN
        DROP TABLE BookShoppingCart;
    END;
GO -- Run the previous command and begins new batch
IF ( EXISTS ( SELECT    *
              FROM      INFORMATION_SCHEMA.TABLES
              WHERE     TABLE_NAME = 'Book' ) )
    BEGIN
        DROP TABLE Book;
    END;
GO -- Run the previous command and begins new batch
```