

A banner image showing a close-up of hands typing on a keyboard. The text 'Hands on...' is overlaid in a large, blue, sans-serif font.

Hands on...

SQL Server SQL CROSS APPLY Statement

In this document we look at:

- [CROSS APPLY](#)
- [OUTER APPLY](#)
- [CROSS APPLY with User Defined Table Valued Functions](#)

The CROSS APPLY statement behaves in a similar fashion to a correlated subquery, but allows us to use ORDER BY statements within the subquery.

The following example produces a list of orders along with the number of days between the order and the next order placed by the same customer.

```
-- Days Between Orders
```

```
USE Northwind
```

```
GO
```

```
SELECT o1.OrderID, o1.OrderDate, ca.OrderID AS NextOrder,
       ca.OrderDate AS NextOrderDate, CustomerID,
       DATEDIFF(DAY, o1.OrderDate, ca.OrderDate) DaysToNextOrder
FROM Orders AS o1
     CROSS APPLY
     (SELECT TOP 1 o.OrderDate, o.OrderID
      FROM Orders AS o
      WHERE o.customerID = o1.customerID
            AND o.OrderID > o1.OrderID
      ORDER BY OrderID) AS ca

ORDER BY CustomerID, o1.OrderID
```

CROSS APPLY enables us to join the order records (**Orders AS o1**) to the subquery (derived table named as **ca**), but we can also use an ORDER BY in the subquery to sort the order records in the subquery in ascending order of orderdate to enable us to identify the first order (TOP 1) after the current record's orderdate.

```

-- Days Between Orders
SELECT o1.OrderID, o1.OrderDate, ca.OrderID AS NextOrder,
      ca.OrderDate AS NextOrderDate, CustomerID,
      DATEDIFF(DAY, o1.OrderDate, ca.OrderDate) DaysToNextOrder
FROM Orders o1
  CROSS APPLY
      (SELECT TOP 1 o.OrderDate, o.OrderID
       FROM Orders AS o WHERE o.customerID = o1.customerID
        AND o.OrderID > o1.OrderID
        ORDER BY OrderID) AS ca
ORDER BY CustomerID, o1.OrderID

```

100 %

Results Messages

	OrderID	OrderDate	NextOrder	NextOrderDate	CustomerID	DaysToNextOrder
1	10643	1997-08-25 00:00:00.000	10692	1997-10-03 00:00:00.000	ALFKI	39
2	10692	1997-10-03 00:00:00.000	10702	1997-10-13 00:00:00.000	ALFKI	10
3	10702	1997-10-13 00:00:00.000	10835	1998-01-15 00:00:00.000	ALFKI	94
4	10835	1998-01-15 00:00:00.000	10952	1998-03-16 00:00:00.000	ALFKI	60
5	10952	1998-03-16 00:00:00.000	11011	1998-04-09 00:00:00.000	ALFKI	24
6	10308	1996-09-18 00:00:00.000	10625	1997-08-08 00:00:00.000	ANATR	324
7	10625	1997-08-08 00:00:00.000	10759	1997-11-28 00:00:00.000	ANATR	112
8	10759	1997-11-28 00:00:00.000	10926	1998-03-04 00:00:00.000	ANATR	96
9	10365	1996-11-27 00:00:00.000	10507	1997-04-15 00:00:00.000	ANTON	139
10	10507	1997-04-15 00:00:00.000	10535	1997-05-13 00:00:00.000	ANTON	28
11	10535	1997-05-13 00:00:00.000	10573	1997-06-19 00:00:00.000	ANTON	37
12	10573	1997-06-19 00:00:00.000	10677	1997-09-22 00:00:00.000	ANTON	95
13	10677	1997-09-22 00:00:00.000	10682	1997-09-25 00:00:00.000	ANTON	3
14	10682	1997-09-25 00:00:00.000	10856	1998-01-28 00:00:00.000	ANTON	125
15	10355	1996-11-15 00:00:00.000	10392	1996-12-16 00:00:00.000	AROUT	31

OUTER APPLY

The OUTER APPLY statement behaves like an OUTER JOIN.

The following example will retain all orders from the outer query (Orders AS o1) even if there are no subsequent orders. The previous version will only display orders that have subsequent orders.

```
-- Days Between Orders - Include orders where there is no further order
USE Northwind
GO
SELECT o1.OrderID, o1.OrderDate, ca.OrderID AS NextOrder,
       ca.OrderDate AS NextOrderDate, CustomerID,
       DATEDIFF(DAY, o1.OrderDate, ca.OrderDate) DaysToNextOrder
FROM Orders AS o1
     OUTER APPLY
       (SELECT TOP 1 o.OrderDate, o.OrderID
        FROM Orders AS o
        WHERE o.customerID = o1.customerID
              AND o.OrderID > o1.OrderID
        ORDER BY OrderID) AS ca
ORDER BY CustomerID, o1.OrderID
```

	OrderID	OrderDate	NextOrder	NextOrderDate	CustomerID	DaysToNextOrder
4	10251	1996-07-08 00:00:00.000	10334	1996-10-21 00:00:00.000	VICTE	105
5	10252	1996-07-09 00:00:00.000	10302	1996-09-10 00:00:00.000	SUPRD	63
6	10253	1996-07-10 00:00:00.000	10541	1997-05-19 00:00:00.000	HANAR	313
7	10254	1996-07-11 00:00:00.000	10370	1996-12-03 00:00:00.000	CHOPS	145
8	10255	1996-07-12 00:00:00.000	10419	1997-01-20 00:00:00.000	RICSU	192
9	10256	1996-07-15 00:00:00.000	10420	1997-01-21 00:00:00.000	WELLI	190
10	10257	1996-07-16 00:00:00.000	10395	1996-12-26 00:00:00.000	HILAA	163
11	10258	1996-07-17 00:00:00.000	10263	1996-07-23 00:00:00.000	ERNSH	6
12	10259	1996-07-18 00:00:00.000	NULL	NULL	CENTC	NULL
13	10260	1996-07-19 00:00:00.000	10407	1997-01-07 00:00:00.000	OTTIK	172
14	10261	1996-07-19 00:00:00.000	10291	1996-08-27 00:00:00.000	QUEDE	39
15	10262	1996-07-22 00:00:00.000	10272	1996-08-02 00:00:00.000	RATTC	11
16	10263	1996-07-23 00:00:00.000	10351	1996-11-11 00:00:00.000	ERNSH	111
17	10264	1996-07-24 00:00:00.000	10327	1996-10-11 00:00:00.000	FOLKO	79
18	10265	1996-07-25 00:00:00.000	10297	1996-09-04 00:00:00.000	BLONP	41
19	10266	1996-07-26 00:00:00.000	10270	1996-08-01 00:00:00.000	WARTH	6
20	10267	1996-07-29 00:00:00.000	10337	1996-10-24 00:00:00.000	FRANK	87
21	10268	1996-07-30 00:00:00.000	10785	1997-12-18 00:00:00.000	GROSR	506

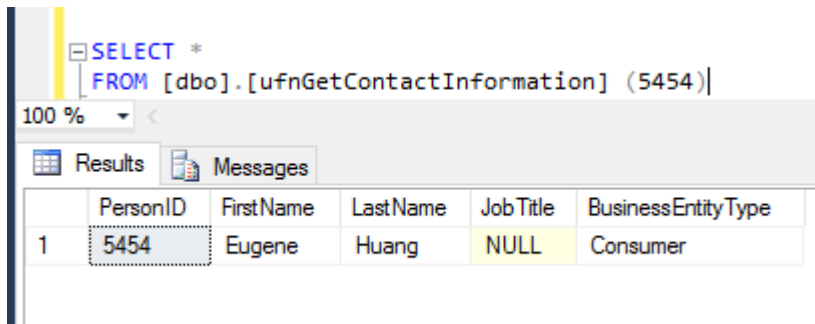
CROSS APPLY and Table Valued Functions

Table Valued Functions return a set of records as output. If we simply wish to view all records returned for a given input value we can call a user defined function as follows:

```
USE AdventureWorks2014
```

```
GO
```

```
SELECT *  
FROM [dbo].[ufnGetContactInformation] (5454)
```



The screenshot shows a SQL query window with the following text:

```
SELECT *  
FROM [dbo].[ufnGetContactInformation] (5454)
```

Below the query window, the 'Results' tab is active, displaying a table with the following data:

	PersonID	FirstName	LastName	JobTitle	BusinessEntityType
1	5454	Eugene	Huang	NULL	Consumer

If, however, we wish to pass a value into the function for each record in another table we need to join the user defined function and the table. This requires a CROSS APPLY statement as we cannot use an INNER JOIN with an ON clause – the join value needs to be passed into the function.

See the next example.

```
USE AdventureWorks2014
```

```
GO
```

```
SELECT AccountNumber, ci.*
FROM Sales.Customer
      CROSS APPLY [dbo].[ufnGetContactInformation] (PersonID) AS ci
```

	AccountNumber	PersonID	FirstName	LastName	JobTitle	BusinessEntityType
1	AW00011000	13531	Jon	Yang	NULL	Consumer
2	AW00011001	5454	Eugene	Huang	NULL	Consumer
3	AW00011002	11269	Ruben	Torres	NULL	Consumer
4	AW00011003	11358	Christy	Zhu	NULL	Consumer
5	AW00011004	11901	Elizabeth	Johnson	NULL	Consumer
6	AW00011005	6990	Julio	Ruiz	NULL	Consumer
7	AW00011006	6229	Janet	Alvarez	NULL	Consumer
8	AW00011007	3878	Marco	Mehta	NULL	Consumer
9	AW00011008	14673	Rob	Verhoff	NULL	Consumer
10	AW00011009	20229	Shannon	Carlson	NULL	Consumer
11	AW00011010	6377	Jacquelyn	Suarez	NULL	Consumer
12	AW00011011	18258	Curtis	Lu	NULL	Consumer
13	AW00011012	11663	Lauren	Walker	NULL	Consumer
14	AW00011013	13873	Ian	Jenkins	NULL	Consumer
15	AW00011014	2868	Sydney	Bennett	NULL	Consumer
16	AW00011015	10963	Chloe	Young	NULL	Consumer
17	AW00011016	3800	Wyatt	Hill	NULL	Consumer

To include all customer records, even those that do not have associated contact information (stores, for example) use an OUTER APPLY:

```
USE AdventureWorks2014
```

```
GO
```

```
SELECT AccountNumber, ci.*
FROM Sales.Customer
      OUTER APPLY [dbo].[ufnGetContactInformation] (PersonID) AS ci
```