(T18)討論Pivot和Unpivot  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T18)討論Pivot和Unpivot  
=======================================================================  
0. Summary

-----------

1. Pivot : HouseSoldRecord1 Table

1.1. Create Sample Data

1.2. GROUP BY

1.3. Pivot need derived table

-----------

2. Pivot : HouseSoldRecord2 Table

2.1. Create Sample Data

2.2. GROUP BY

2.3. Logic Error : Pivot need derived table

2.4. Pivot need derived table

2.5. 3 columns in derived table

2.6. 4 columns in derived table

2.7. 6 columns in derived table

-----------

3. Pivot : HouseSoldRecord3 Table

3.1. Create Sample Data

3.2. GROUP BY

3.3. 2 columns in derived table

3.4. 3 columns in derived table

3.5. 4 columns in derived table

3.6. 6 columns in derived table

3.7. dynamic sql query

3.7.1. fnGetAllSuburb

3.7.2. sp\_executesql

-----------

4. Clean up

-----------

5. PIVOT\_UNPIVOT : HouseSoldRecord4 Table

5.1. Create Sample Data

5.2. Pivot need derived table

5.3. The following clauses are equivalent

-----------

6. PIVOT\_UNPIVOT : HouseSoldRecord5 Table

6.1. Create Sample Data

6.2. Pivot need derived table

6.3. The following clauses are NOT equivalent

-----------

7. Clean up  
=======================================================================

0. Summary

1.

Pivot Syntax1

Reference:

[https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/ms177410%28v=sql.105%29.aspx)

--SELECT  \*

--FROM    --derived table

--        ( SELECT    T1C1 , --1st pivoted column

--                    T1C2 , --2nd pivoted column

--                    ....

--                    T1Cn-2 , --N-2 th pivoted column

--                    T1Cn-1 , --Column n-1 that contains the values that will become column headers

--                    T1Cn AS T1CnAliasName --Column n that used for aggregation function

--          FROM      T1

--      ) AS BaseData PIVOT

--      ( SUM(T1CnAliasName) FOR T1Cn-2 IN ( T1Cn-1V1, T1Cn-1V2, T1Cn-1V3 ) ) AS PivotTable;

--ORDER BY T1C1, T1C2, ... , T1Cn-2

--GO -- Run the previous command and begins new batch

T stand for Table

C stand for column

V stand for Value

T1C1, T1C2, ... , T1Cn-2 will become the pivoted columns in left hand side.

Column n-1,T1Cn-1, that contains the values that will become column headers.

Column n,T1Cn, that used for aggregation function.

Using "ORDER BY T1Cn-1V1, T1Cn-1V2, T1Cn-1V3, T1Cn" might cause some logic error that we don't expect.

Better just use "ORDER BY T1C1, T1C2, ... , T1Cn-2".

---------------------------

1.1.

E.g.

--SELECT  \*

--FROM    --derived table

--        ( SELECT    AgentName ,    --1st pivoted column

--                    SoldSuburb ,    --Column n-1 that contains the values that will become column headers

--                    SoldPrice AS TotalSales    --Column n that used for aggregation function

--          FROM      HouseSoldRecord2

--        ) AS BaseData PIVOT

--        ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

--GO -- Run the previous command and begins new batch

AgentName will become the pivoted columns in left hand side.

Column SoldSuburb that contains the values that will become column headers, Suburb01, Suburb02, Suburb03.

Column SoldPrice that used for aggregation function.

Using "ORDER BY Suburb01, Suburb02, Suburb03, SoldPrice" might cause some logic error that we don't expect.

Better just use "ORDER BY AgentName".

------------------------------------------------------------

2.

Pivot Syntax2

Reference:

[https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/ms177410%28v=sql.105%29.aspx)

If you don't want to display all pivoted columns in left hand side.

Then you cannot use "SELECT \*",

you have to use "SELECT T1C1, T1C2, ..." in outter query.

--SELECT  T1C1, T1C2, ...T1Cn-3, T1Cn-1V1, T1Cn-1V2, T1Cn-1V3

---- You hide T1Cn-2 pivoted column, but You still can ORDER BY T1Cn-2.

--FROM    --derived table

--        ( SELECT    T1C1 , --1st pivoted column

--                    T1C2 , --2nd pivoted column

--                    ....

--                    T1Cn-3 , --N-3 th pivoted column

--                    T1Cn-2 , --N-2 th pivoted column which you don't want to display.

--                    T1Cn-1 , --Column n-1 that contains the values that will become column headers

--                    T1Cn AS T1CnAliasName --Column n that used for aggregation function

--          FROM      T1

--      ) AS BaseData PIVOT

--      ( SUM(T1CnAliasName) FOR T1Cn-2 IN ( T1Cn-1V1, T1Cn-1V2, T1Cn-1V3 ) ) AS PivotTable;

--ORDER BY T1C1, T1C2, ... , T1Cn-2

--GO -- Run the previous command and begins new batch

T stand for Table

C stand for column

V stand for Value

T1C1, T1C2, ... , T1Cn-3 will become the pivoted columns in left hand side.

You hide T1Cn-2 pivoted column, but You still can ORDER BY T1Cn-2.

Column n-1,T1Cn-1, that contains the values that will become column headers.

Column n,T1Cn, that used for aggregation function.

Using "ORDER BY T1Cn-1V1, T1Cn-1V2, T1Cn-1V3, T1Cn" might cause some logic error that we don't expect.

Better just use "ORDER BY T1C1, T1C2, ... , T1Cn-2".

---------------------------

2.1.

E.g.

--SELECT  AgentName, SoldYear, SoldMonthName, Suburb01, Suburb02, Suburb03

--FROM    --derived table

--        ( SELECT    AgentName ,

--                    YEAR(SoldDateTime) AS SoldYear ,

--                    DATEPART(MONTH, SoldDateTime) AS SoldMonth ,

--                      DATENAME(MM,SoldDateTime) AS SoldMonthName ,

--                    SoldSuburb ,

--                    SoldPrice AS TotalSales

--          FROM      HouseSoldRecord2

--        ) AS BaseData PIVOT

--        ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable

--ORDER BY AgentName , SoldYear, SoldMonth

--GO -- Run the previous command and begins new batch

2.1.1.

AgentName, SoldYear, SoldMonthName will become the pivoted columns in left hand side.

Column SoldSuburb that contains the values that will become column headers, Suburb01, Suburb02, Suburb03.

Column SoldPrice that used for aggregation function.

Using "ORDER BY Suburb01, Suburb02, Suburb03, SoldSuburb" might cause some logic error that we don't expect.

Better just use "AgentName, SoldYear, SoldMonth".

2.1.2.

If using "ORDER BY AgentName , SoldYear, SoldMonthName"

SoldMonthName will become the alphabet order.

E.g. "April", "December", "July", "June" ,"November" ...etc.

This is not what we want,

we want the order by SoldMonth number but display SoldMonthName

E.g. "April", ... , "June", "July",..., "November" ,"December"

Thus, inner derived table SELECT both  SoldMonth and SoldMonthName.

But outer pivot table only SELECT SoldMonthName.

In addition, we still can ORDER BY SoldMonth.

------------------------------------------------------------

3.

Pivot and UnPivot Syntax3

3.1.

If the PIVOT operator has not aggregated the data,

you can get your original data back using the UNPIVOT operator

but If the PIVOT operator has aggregated the data,

then you can NOT use UNPIVOT operator.

3.2.

3.2.1.

Create PIVOT Table View Syntax.

--IF ( EXISTS ( SELECT    \*

--              FROM      INFORMATION\_SCHEMA.TABLES

--              WHERE     TABLE\_NAME = 'vwName' ) )

--    BEGIN

--        DROP VIEW vwName;

--    END;

--GO -- Run the previous command and begins new batch

--CREATE VIEW vwName

--AS

--    --SELECT PIVOT Table ...

--GO -- Run the previous command and begins new batch

----See the View data

--SELECT  \*

--FROM    vwName;

--GO -- Run the previous command and begins new batch

-----------------------------------

3.2.2.

E.g.

----Delete View if exist

--IF ( EXISTS ( SELECT    \*

--              FROM      INFORMATION\_SCHEMA.TABLES

--              WHERE     TABLE\_NAME = 'vwHouseSoldRecord5Pivot1' ) )

--    BEGIN

--        DROP VIEW vwHouseSoldRecord5Pivot1;

--    END;

--GO -- Run the previous command and begins new batch

----Create view for HouseSoldRecord5 Povit Table

--CREATE VIEW vwHouseSoldRecord5Pivot1

--AS

--    SELECT  AgentName ,

--            Suburb01 ,

--            Suburb02 ,

--            Suburb03

--    FROM    HouseSoldRecord5 PIVOT

--( SUM(SoldPrice) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

--GO -- Run the previous command and begins new batch

----See the View data

--SELECT  \*

--FROM    vwHouseSoldRecord5Pivot1;

--GO -- Run the previous command and begins new batch

-----------------------------------

3.3.

3.3.1.

UnPivot from PIVOT Table View Syntax.

--SELECT C1, C2, C3

--FROM vwName

--UNPIVOT

--(

--       C3

--       FOR C2 IN (C2V1, C2V2 ,C2V3)

--) AS UnpivotExample

--ORDER BY C1, C2;

--GO -- Run the previous command and begins new batch

-----------------------------------

3.3.2.

E.g.

--SELECT AgentName, SoldSuburb, SoldPrice

--FROM vwHouseSoldRecord5Pivot1

--UNPIVOT

--(

--       SoldPrice

--       FOR SoldSuburb IN (Suburb01, Suburb02 ,Suburb03)

--) AS UnpivotExample

--ORDER BY AgentName, SoldSuburb;

--GO -- Run the previous command and begins new batch

==================================================

1. Pivot : HouseSoldRecord1 Table

--================================================================

--T018\_01\_Pivot : HouseSoldRecord1 Table

--================================================================

1.1. Create Sample Data

--================================================================

--T018\_01\_01

--HouseSoldRecord1 Table

--Create Sample Data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord1' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord1;

        DROP TABLE HouseSoldRecord1;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSoldRecord1

    (

      AgentName NVARCHAR(100) ,

      SoldSuburb NVARCHAR(100) ,

      SoldPrice MONEY

    );

GO -- Run the previous command and begins new batch

INSERT  HouseSoldRecord1

VALUES  ( N'Name01', N'Suburb02', 400000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name02', N'Suburb01', 500000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb01', 560000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name02', N'Suburb02', 350000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb02', 440000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb03', 460000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb03', 470000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name02', N'Suburb01', 330000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name01', N'Suburb01', 470000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb03', 320000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name01', N'Suburb01', 390000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name02', N'Suburb02', 350000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb03', 430000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name02', N'Suburb03', 440000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb02', 450000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb01', 475000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name03', N'Suburb02', 489000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name02', N'Suburb02', 399000 );

INSERT  HouseSoldRecord1

VALUES  ( N'Name01', N'Suburb03', 499000 );

INSERT  HouseSoldRecord1

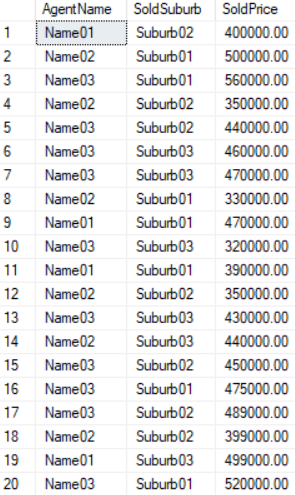
VALUES  ( N'Name03', N'Suburb01', 520000 );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    HouseSoldRecord1;

GO -- Run the previous command and begins new batch



1.2. GROUP BY

--================================================================

--T018\_01\_02

--HouseSoldRecord1 Table

--GROUP BY

SELECT  SoldSuburb ,

        AgentName ,

        SUM(SoldPrice) AS Total

FROM    HouseSoldRecord1

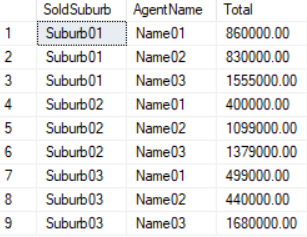
GROUP BY SoldSuburb ,

        AgentName

ORDER BY SoldSuburb ,

        AgentName;

GO -- Run the previous command and begins new batch



1.3. Pivot need derived table

--================================================================

--T018\_01\_03

--HouseSoldRecord1 Table

--Pivot need derived table

SELECT  AgentName ,

        Suburb01 ,

        Suburb02 ,

        Suburb03

FROM    HouseSoldRecord1 PIVOT

( SUM(SoldPrice) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

/\*

Output as the following

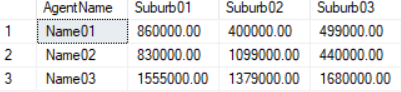
--AgentName Suburb01   Suburb02   Suburb03

--Name01    860000.00  400000.00  499000.00

--Name02    830000.00  1099000.00 440000.00

--Name03    1555000.00 1379000.00 1680000.00

\*/



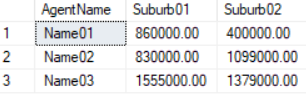
SELECT  AgentName ,

        Suburb01 ,

        Suburb02

FROM    HouseSoldRecord1 PIVOT

( SUM(SoldPrice) FOR SoldSuburb IN ( Suburb01, Suburb02 ) ) AS PivotTable;



/\*

1.

Output as the following

--AgentName Suburb01   Suburb02

--Name01    860000.00  400000.00

--Name02    830000.00  1099000.00

--Name03    1555000.00 1379000.00

2.

Pivot need derived table

2.1.

The PIVOT query for HouseSoldRecord1 converts the unique column values (Suburb01, Suburb02, Suburb03)

in SoldSuburb column into Columns in the output,

along with performing aggregations on the SoldPrice column.

The Outer query, simply, selects AgentName column from HouseSoldRecord1 table,

along with pivoted columns from the PivotTable.

2.2.

In real world, Table should have any number of columns.

However,

HouseSoldRecord1 only has 3 columns, AgentName, SoldSuburb, and SoldPrice.

Not every table only has 3 columns.

There will be a 'logic error' if the table has more than 3 columns.

Let's try it.

\*/

==================================================

2. Pivot : HouseSoldRecord2 Table

--================================================================

--T018\_02\_Pivot : HouseSoldRecord2 Table

--================================================================

2.1. Create Sample Data

--================================================================

--T018\_02\_01

--Create Sample Data

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord2' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord2;

        DROP TABLE HouseSoldRecord2;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSoldRecord2

    (

      Id INT IDENTITY(1, 1)

             PRIMARY KEY ,

      AgentName NVARCHAR(100) ,

      SoldSuburb NVARCHAR(100) ,

      SoldPrice MONEY ,

      SoldDateTime DATETIME NULL

    );

GO -- Run the previous command and begins new batch

INSERT  HouseSoldRecord2

VALUES  ( N'Name01', N'Suburb02', 400000,

          CAST(N'2016-04-12 13:27:58.600' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name02', N'Suburb01', 500000,

          CAST(N'2017-04-02 13:53:29.587' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb01', 560000,

          CAST(N'2015-09-01 00:22:21.050' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name02', N'Suburb02', 350000,

          CAST(N'2015-09-16 07:20:09.037' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb02', 440000,

          CAST(N'2016-01-31 00:59:21.860' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb03', 460000,

          CAST(N'2016-04-19 07:12:38.813' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb03', 470000,

          CAST(N'2017-04-02 09:06:19.740' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name02', N'Suburb01', 330000,

          CAST(N'2017-03-01 16:25:42.177' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name01', N'Suburb01', 470000,

          CAST(N'2015-04-13 21:02:58.543' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb03', 320000,

          CAST(N'2016-07-04 17:55:15.250' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name01', N'Suburb01', 390000,

          CAST(N'2016-12-27 13:01:05.440' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name02', N'Suburb02', 350000,

          CAST(N'2016-08-30 04:21:14.810' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb03', 430000,

          CAST(N'2015-07-31 02:17:26.717' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name02', N'Suburb03', 440000,

          CAST(N'2016-06-15 15:26:28.500' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb02', 450000,

          CAST(N'2017-04-09 01:24:11.440' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb01', 475000,

          CAST(N'2015-02-26 00:39:14.323' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb02', 489000,

          CAST(N'2015-08-28 04:50:27.180' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name02', N'Suburb02', 399000,

          CAST(N'2016-11-07 00:48:09.930' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name01', N'Suburb03', 499000,

          CAST(N'2015-11-15 09:40:58.647' AS DATETIME) );

INSERT  HouseSoldRecord2

VALUES  ( N'Name03', N'Suburb01', 520000,

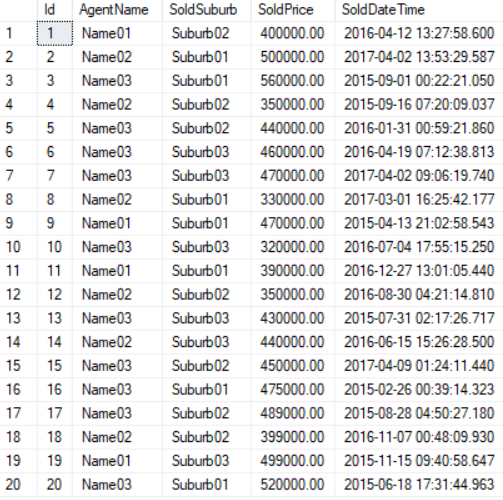
          CAST(N'2015-06-18 17:31:44.963' AS DATETIME) );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    HouseSoldRecord2;

GO -- Run the previous command and begins new batch



2.2. GROUP BY

--================================================================

--T018\_02\_02

--HouseSoldRecord2 Table

--GROUP BY

SELECT  SoldSuburb ,

        AgentName ,

        SUM(SoldPrice) AS Total

FROM    HouseSoldRecord2

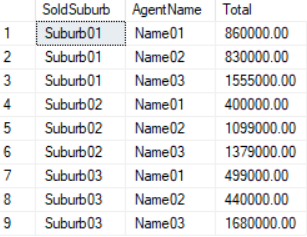
GROUP BY SoldSuburb ,

        AgentName

ORDER BY SoldSuburb ,

        AgentName;

GO -- Run the previous command and begins new batch



2.3. Logic Error : Pivot need derived table

--================================================================

--T018\_02\_03

--HouseSoldRecord2 Table

--Logic Error : Pivot need derived table

SELECT  AgentName ,

        Suburb01 ,

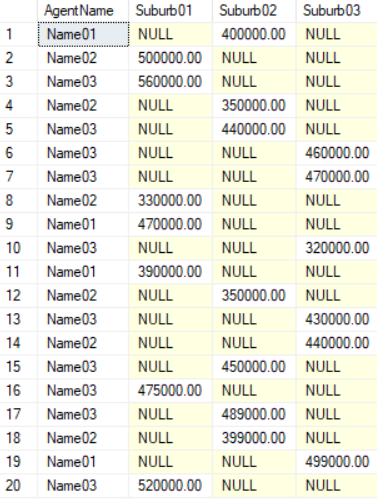
        Suburb02 ,

        Suburb03

FROM    HouseSoldRecord2 PIVOT

( SUM(SoldPrice) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch



/\*

1.

Logic Error.

IOutput as the following

--AgentName Suburb01   Suburb02   Suburb03

--Name01     NULL   400000.00    NULL

--Name02     500000.00    NULL   NULL

...

--Name02     NULL   399000.00    NULL

--Name01     NULL   NULL   499000.00

--Name03     520000.00    NULL   NULL

Total 20 rows.

This is not what we expect.

2.

Pivot need derived table

2.1.

HouseSoldRecord2 has 5 columns,

Id, AgentName, SoldSuburb, SoldPrice MONEY, and SoldDateTime.

This is because of the presence of Id and SoldDateTime column in HouseSoldRecord2,

which is also considered when performing pivoting and group by.

To eliminate this from the calculations,

we have used derived table, which only selects,

AgentName, SoldSuburb, and SoldPrice.

The rest of the query is very similar to what we have already seen.

\*/

2.4. Pivot need derived table

--================================================================

--T018\_02\_04

--HouseSoldRecord2 Table

--Pivot need derived table

-----------------------------------------------------------

--T018\_02\_04\_01 : HouseSoldRecord2 Table

--Pivot need derived table, the following clauses are equivalent:

--2 columns in derived table : SoldSuburb, SoldPrice

-----------------------------------------

--T018\_02\_04\_01\_01

SELECT  Suburb01 ,

        Suburb02 ,

        Suburb03

FROM    --derived table

        ( SELECT    SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      HouseSoldRecord2

        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch

-----------------------------------------

--T018\_02\_04\_01\_02

SELECT  \*

FROM    --derived table

        ( SELECT    SoldSuburb ,

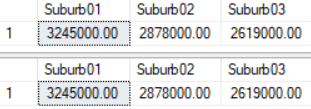
                    SoldPrice AS TotalSales

          FROM      HouseSoldRecord2

        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch



/\*

1.

1.1.

2 columns in derived table : SoldSuburb, SoldPrice

Output as the following

--Suburb01   Suburb02   Suburb03

--3245000.00 2878000.00 2619000.00

1.2.

The only different is the following.

1.2.1.

1st Query

--SELECT  Suburb01 ,

--        Suburb02 ,

--        Suburb03

1.2.2.

2nd Query

--SELECT  \*

we can always replace

outter "SELECT C1,C2...etc" by "SELECT  \*".

The inner query "SELECT C1,C2...etc" is more important

which decide how many columns pivot to left side.

2.

Let's see next sample to conclude the Pivot Syntax.

\*/

2.5. 3 columns in derived table

--================================================================

--T018\_02\_05 : HouseSoldRecord2 Table

--Pivot need derived table, the following clauses are equivalent:

--3 columns in derived table : AgentName, SoldSuburb, SoldPrice

-----------------------------------------

--T018\_02\_05\_01

SELECT  AgentName ,

        Suburb01 ,

        Suburb02 ,

        Suburb03

FROM    --derived table

        ( SELECT    AgentName ,

                    SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      HouseSoldRecord2

        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch

-----------------------------------------

--T018\_02\_05\_02

SELECT  \*

FROM    --derived table

        ( SELECT    AgentName ,

                    SoldSuburb ,

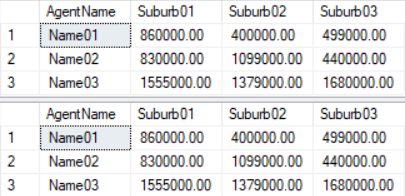
                    SoldPrice AS TotalSales

          FROM      HouseSoldRecord2

        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch



/\*

1.

1.1.

3 columns in derived table : AgentName, SoldSuburb, SoldPrice

Output as the following

--AgentName Suburb01   Suburb02   Suburb03

--Name01    860000.00  400000.00  499000.00

--Name02    830000.00  1099000.00 440000.00

--Name03    1555000.00 1379000.00 1680000.00

1.2.

The only different is the following.

1.2.1.

1st Query

--SELECT  AgentName ,

--        Suburb01 ,

--        Suburb02 ,

--        Suburb03

1.2.2.

2nd Query

--SELECT  \*

we can always replace

outter "SELECT C1,C2...etc" by "SELECT  \*".

The inner query "SELECT C1,C2...etc" is more important

which decide how many columns pivot to left side.

-----------------

2.

Pivot Syntax1

Reference:

[https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/ms177410%28v=sql.105%29.aspx)

--SELECT  \*

--FROM    --derived table

--        ( SELECT    T1C1 , --1st pivoted column

--                    T1C2 , --2nd pivoted column

--                    ....

--                    T1Cn-2 , --N-2 th pivoted column

--                    T1Cn-1 , --Column n-1 that contains the values that will become column headers

--                    T1Cn AS T1CnAliasName --Column n that used for aggregation function

--          FROM      T1

--      ) AS BaseData PIVOT

--      ( SUM(T1CnAliasName) FOR T1Cn-2 IN ( T1Cn-1V1, T1Cn-1V2, T1Cn-1V3 ) ) AS PivotTable;

--ORDER BY T1C1, T1C2, ... , T1Cn-2

--GO -- Run the previous command and begins new batch

T stand for Table

C stand for column

V stand for Value

T1C1, T1C2, ... , T1Cn-2 will become the pivoted columns in left hand side.

Column n-1,T1Cn-1, that contains the values that will become column headers.

Column n,T1Cn, that used for aggregation function.

Using "ORDER BY T1Cn-1V1, T1Cn-1V2, T1Cn-1V3, T1Cn" might cause some logic error that we don't expect.

Better just use "ORDER BY T1C1, T1C2, ... , T1Cn-2".

---------------------------

2.1.

E.g.

--SELECT  \*

--FROM    --derived table

--        ( SELECT    AgentName ,--1st pivoted column

--                    SoldSuburb ,      --Column n-1 that contains the values that will become column headers

--                    SoldPrice AS TotalSales  --Column n that used for aggregation function

--          FROM      HouseSoldRecord2

--        ) AS BaseData PIVOT

--           ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

--GO -- Run the previous command and begins new batch

AgentName will become the pivoted columns in left hand side.

Column SoldSuburb that contains the values that will become column headers, Suburb01, Suburb02, Suburb03.

Column SoldPrice that used for aggregation function.

Using "ORDER BY Suburb01, Suburb02, Suburb03, SoldPrice" might cause some logic error that we don't expect.

Better just use "ORDER BY AgentName".

\*/

2.6. 4 columns in derived table

--================================================================

--T018\_02\_06

--HouseSoldRecord2 Table

--Pivot need derived table

--4 columns in derived table : AgentName, YEAR(SoldDateTime), SoldSuburb, SoldPrice

SELECT  \*

FROM    --derived table

        ( SELECT    AgentName ,

                    YEAR(SoldDateTime) AS SoldYear ,

                    SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      HouseSoldRecord2

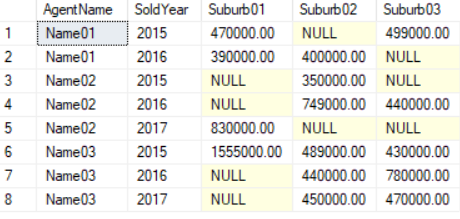
        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable

ORDER BY AgentName ,

        SoldYear;

GO -- Run the previous command and begins new batch



2.7. 6 columns in derived table

--================================================================

--T018\_02\_07

--HouseSoldRecord2 Table

--Pivot need derived table

--6 columns in derived table :

--AgentName, YEAR(SoldDateTime), DATEPART(MONTH, SoldDateTime), DATENAME(MM,SoldDateTime), SoldSuburb, SoldPrice

------------------------------------

--T018\_02\_07\_01

SELECT  \*

FROM    --derived table

        ( SELECT    AgentName ,

                    YEAR(SoldDateTime) AS SoldYear ,

                    DATENAME(MM, SoldDateTime) AS SoldMonthName ,

                    SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      HouseSoldRecord2

        ) AS BaseData PIVOT

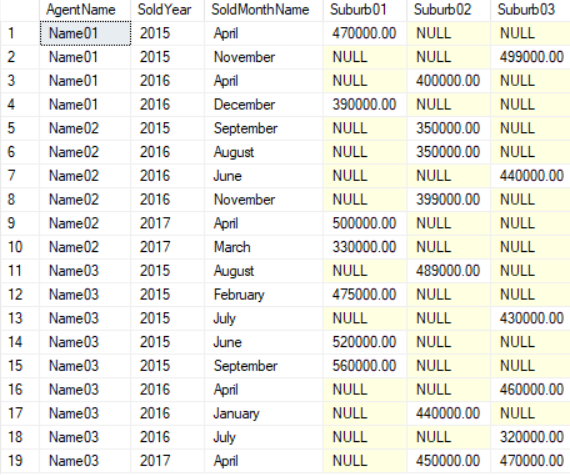
             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable

ORDER BY AgentName ,

        SoldYear ,

        SoldMonthName;

GO -- Run the previous command and begins new batch



------------------------------------

--T018\_02\_07\_02

SELECT  AgentName ,

        SoldYear ,

        SoldMonthName ,

        Suburb01 ,

        Suburb02 ,

        Suburb03

FROM    --derived table

        ( SELECT    AgentName ,

                    YEAR(SoldDateTime) AS SoldYear ,

                    DATEPART(MONTH, SoldDateTime) AS SoldMonth ,

                    DATENAME(MM, SoldDateTime) AS SoldMonthName ,

                    SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      HouseSoldRecord2

        ) AS BaseData PIVOT

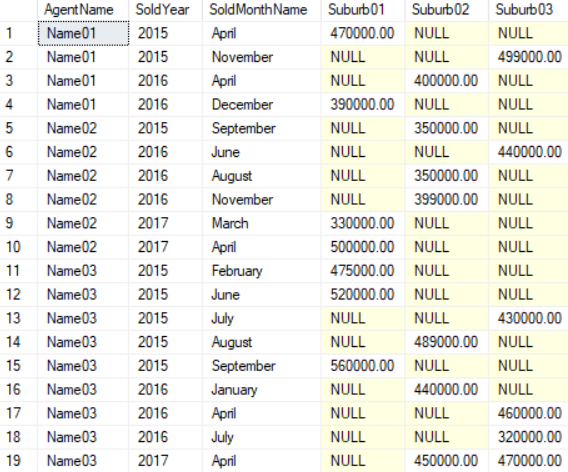
             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable

ORDER BY AgentName ,

        SoldYear ,

        SoldMonth;

GO -- Run the previous command and begins new batch



/\*

1.

Pivot Syntax2

Reference:

[https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/ms177410%28v=sql.105%29.aspx)

If you don't want to display all pivoted columns in left hand side.

Then you cannot use "SELECT \*",

you have to use "SELECT T1C1, T1C2, ..." in outter query.

--SELECT  T1C1, T1C2, ...T1Cn-3, T1Cn-1V1, T1Cn-1V2, T1Cn-1V3

---- You hide T1Cn-2 pivoted column, but You still can ORDER BY T1Cn-2.

--FROM    --derived table

--        ( SELECT    T1C1 , --1st pivoted column

--                    T1C2 , --2nd pivoted column

--                    ....

--                    T1Cn-3 , --N-3 th pivoted column

--                    T1Cn-2 , --N-2 th pivoted column which you don't want to display.

--                    T1Cn-1 , --Column n-1 that contains the values that will become column headers

--                    T1Cn AS T1CnAliasName --Column n that used for aggregation function

--          FROM      T1

--      ) AS BaseData PIVOT

--      ( SUM(T1CnAliasName) FOR T1Cn-2 IN ( T1Cn-1V1, T1Cn-1V2, T1Cn-1V3 ) ) AS PivotTable;

--ORDER BY T1C1, T1C2, ... , T1Cn-2

--GO -- Run the previous command and begins new batch

T stand for Table

C stand for column

V stand for Value

T1C1, T1C2, ... , T1Cn-3 will become the pivoted columns in left hand side.

You hide T1Cn-2 pivoted column, but You still can ORDER BY T1Cn-2.

Column n-1,T1Cn-1, that contains the values that will become column headers.

Column n,T1Cn, that used for aggregation function.

Using "ORDER BY T1Cn-1V1, T1Cn-1V2, T1Cn-1V3, T1Cn" might cause some logic error that we don't expect.

Better just use "ORDER BY T1C1, T1C2, ... , T1Cn-2".

---------------------------

1.1.

E.g.

--SELECT  AgentName, SoldYear, SoldMonthName, Suburb01, Suburb02, Suburb03

--FROM    --derived table

--        ( SELECT    AgentName ,

--                    YEAR(SoldDateTime) AS SoldYear ,

--                    DATEPART(MONTH, SoldDateTime) AS SoldMonth ,

--                                 DATENAME(MM,SoldDateTime) AS SoldMonthName ,

--                    SoldSuburb ,

--                    SoldPrice AS TotalSales

--          FROM      HouseSoldRecord2

--        ) AS BaseData PIVOT

--           ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable

--ORDER BY AgentName , SoldYear, SoldMonth

--GO -- Run the previous command and begins new batch

2.1.1.

AgentName, SoldYear, SoldMonthName will become the pivoted columns in left hand side.

Column SoldSuburb that contains the values that will become column headers, Suburb01, Suburb02, Suburb03.

Column SoldPrice that used for aggregation function.

Using "ORDER BY Suburb01, Suburb02, Suburb03, SoldSuburb" might cause some logic error that we don't expect.

Better just use "AgentName, SoldYear, SoldMonth".

2.1.2.

If using "ORDER BY AgentName , SoldYear, SoldMonthName"

SoldMonthName will become the alphabet order.

E.g. "April", "December", "July", "June" ,"November" ...etc.

This is not what we want,

we want the order by SoldMonth number but display SoldMonthName

E.g. "April", ... , "June", "July",..., "November" ,"December"

Thus, inner derived table SELECT both  SoldMonth and SoldMonthName.

But outer pivot table only SELECT SoldMonthName.

In addition, we still can ORDER BY SoldMonth.

\*/

3. Pivot : HouseSoldRecord3 Table

--================================================================

--T018\_03\_Pivot : HouseSoldRecord3 Table

--================================================================

3.1. Create Sample Data

--================================================================

--T018\_03\_01

--Create Sample Data

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord3' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord3;

        DROP TABLE HouseSoldRecord3;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Suburb3' ) )

    BEGIN

        TRUNCATE TABLE dbo.Suburb3;

        DROP TABLE Suburb3;

    END;

GO -- Run the previous command and begins new batch

-----------------------------

CREATE TABLE Suburb3

    (

      SuburbId INT IDENTITY(1, 1)

                   PRIMARY KEY ,

      SuburbName NVARCHAR(100),

    );

GO -- Run the previous command and begins new batch

INSERT  Suburb3

VALUES  ( N'Suburb01' );

INSERT  Suburb3

VALUES  ( N'Suburb02' );

INSERT  Suburb3

VALUES  ( N'Suburb03' );

GO -- Run the previous command and begins new batch

-----------------------------

CREATE TABLE HouseSoldRecord3

    (

      Id INT IDENTITY(1, 1)

             PRIMARY KEY ,

      AgentName NVARCHAR(100) ,

      HouseSoldSuburbID INT FOREIGN KEY REFERENCES Suburb3 ( SuburbId ) ,

      SoldPrice MONEY ,

      SoldDateTime DATETIME NULL

    );

GO -- Run the previous command and begins new batch

INSERT  HouseSoldRecord3

VALUES  ( N'Name01', 2, 400000, CAST(N'2016-04-12 13:27:58.600' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name02', 1, 500000, CAST(N'2017-04-02 13:53:29.587' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 1, 560000, CAST(N'2015-09-01 00:22:21.050' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name02', 2, 350000, CAST(N'2015-09-16 07:20:09.037' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 2, 440000, CAST(N'2016-01-31 00:59:21.860' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 3, 460000, CAST(N'2016-04-19 07:12:38.813' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 3, 470000, CAST(N'2017-04-02 09:06:19.740' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name02', 1, 330000, CAST(N'2017-03-01 16:25:42.177' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name01', 1, 470000, CAST(N'2015-04-13 21:02:58.543' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 3, 320000, CAST(N'2016-07-04 17:55:15.250' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name01', 1, 390000, CAST(N'2016-12-27 13:01:05.440' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name02', 2, 350000, CAST(N'2016-08-30 04:21:14.810' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 3, 430000, CAST(N'2015-07-31 02:17:26.717' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name02', 3, 440000, CAST(N'2016-06-15 15:26:28.500' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 2, 450000, CAST(N'2017-04-09 01:24:11.440' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 1, 475000, CAST(N'2015-02-26 00:39:14.323' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 2, 489000, CAST(N'2015-08-28 04:50:27.180' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name02', 2, 399000, CAST(N'2016-11-07 00:48:09.930' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name01', 3, 499000, CAST(N'2015-11-15 09:40:58.647' AS DATETIME) );

INSERT  HouseSoldRecord3

VALUES  ( N'Name03', 1, 520000, CAST(N'2015-06-18 17:31:44.963' AS DATETIME) );

GO -- Run the previous command and begins new batch

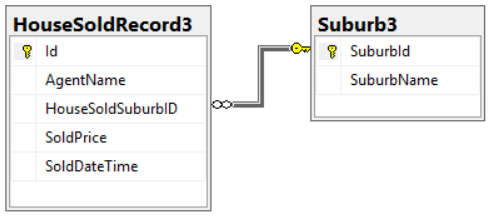
SELECT  \*

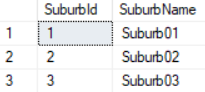
FROM    Suburb3;

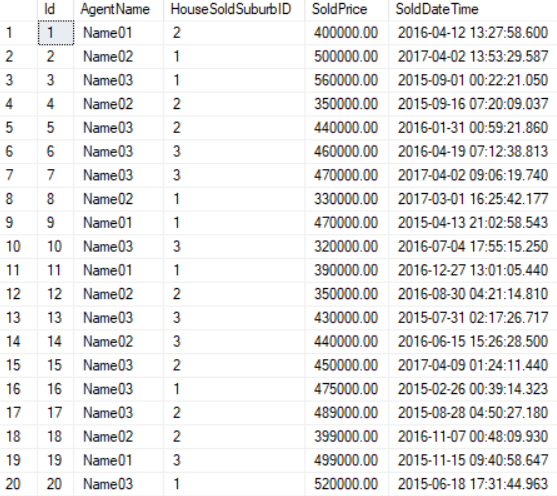
SELECT  \*

FROM    HouseSoldRecord3;

GO -- Run the previous command and begins new batch







3.2. GROUP BY

--================================================================

--T018\_03\_02

--HouseSoldRecord3 Table

--GROUP BY

SELECT  s3.SuburbName ,

        hsr3.AgentName ,

        SUM(hsr3.SoldPrice) AS TotalSales

FROM    Suburb3 s3

        INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

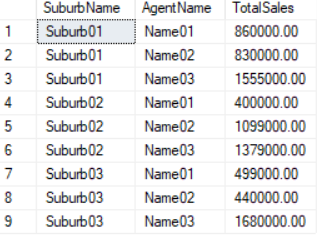
GROUP BY s3.SuburbName ,

        hsr3.AgentName

ORDER BY s3.SuburbName ,

        hsr3.AgentName;

GO -- Run the previous command and begins new batch



3.3. 2 columns in derived table

--================================================================

--T018\_03\_03

--HouseSoldRecord2 Table

--Pivot need derived table

--2 columns in derived table : SoldSuburb, SoldPrice

SELECT  \*

FROM    --derived table

        ( SELECT    s3.SuburbName AS SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      Suburb3 s3

                    INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch

/\*

2 columns in derived table : SoldSuburb, SoldPrice

Output as the following

--Suburb01   Suburb02   Suburb03

--3245000.00 2878000.00 2619000.00

\*/



3.4. 3 columns in derived table

--================================================================

--T018\_03\_04

--HouseSoldRecord3 Table

--Pivot need derived table

--3 columns in derived table : SoldSuburb, SoldPrice

SELECT  \*

FROM    --derived table

        ( SELECT    hsr3.AgentName ,

                    s3.SuburbName AS SoldSuburb ,

                    SoldPrice AS TotalSales

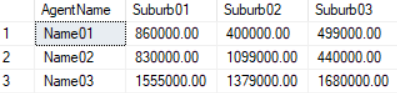
          FROM      Suburb3 s3

                    INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch



/\*

1.

3 columns in derived table : AgentName, SoldSuburb, SoldPrice

Output as the following

--AgentName Suburb01   Suburb02   Suburb03

--Name01    860000.00  400000.00  499000.00

--Name02    830000.00  1099000.00 440000.00

--Name03    1555000.00 1379000.00 1680000.00

-----------------

2.

Pivot Syntax1

Reference:

[https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/ms177410%28v=sql.105%29.aspx)

--SELECT  \*

--FROM    --derived table

--        ( SELECT    T1C1 , --1st pivoted column

--                    T1C2 , --2nd pivoted column

--                    ....

--                    T1Cn-2 , --N-2 th pivoted column

--                    T1Cn-1 , --Column n-1 that contains the values that will become column headers

--                    T1Cn AS T1CnAliasName --Column n that used for aggregation function

--          FROM      T1

--      ) AS BaseData PIVOT

--      ( SUM(T1CnAliasName) FOR T1Cn-2 IN ( T1Cn-1V1, T1Cn-1V2, T1Cn-1V3 ) ) AS PivotTable;

--ORDER BY T1C1, T1C2, ... , T1Cn-2

--GO -- Run the previous command and begins new batch

T stand for Table

C stand for column

V stand for Value

T1C1, T1C2, ... , T1Cn-2 will become the pivoted columns in left hand side.

Column n-1,T1Cn-1, that contains the values that will become column headers.

Column n,T1Cn, that used for aggregation function.

Using "ORDER BY T1Cn-1V1, T1Cn-1V2, T1Cn-1V3, T1Cn" might cause some logic error that we don't expect.

Better just use "ORDER BY T1C1, T1C2, ... , T1Cn-2".

---------------------------

2.1.

E.g.

--SELECT  \*

--FROM    --derived table

--        ( SELECT    hsr3.AgentName ,

--                    s3.SuburbName AS SoldSuburb,

--                    SoldPrice AS TotalSales

--          FROM      Suburb3 s3

--                    INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

--        ) AS BaseData PIVOT

--           ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

--GO -- Run the previous command and begins new batch

AgentName will become the pivoted columns in left hand side.

Column SoldSuburb that contains the values that will become column headers, Suburb01, Suburb02, Suburb03.

Column SoldPrice that used for aggregation function.

Using "ORDER BY Suburb01, Suburb02, Suburb03, SoldPrice" might cause some logic error that we don't expect.

Better just use "ORDER BY AgentName".

\*/

3.5. 4 columns in derived table

--================================================================

--T018\_03\_05

--HouseSoldRecord3 Table

--Pivot need derived table

--4 columns in derived table : AgentName, YEAR(SoldDateTime), SoldSuburb, SoldPrice

SELECT  \*

FROM    --derived table

        ( SELECT    hsr3.AgentName ,

                    YEAR(hsr3.SoldDateTime) AS SoldYear ,

                    s3.SuburbName AS SoldSuburb ,

                    SoldPrice AS TotalSales

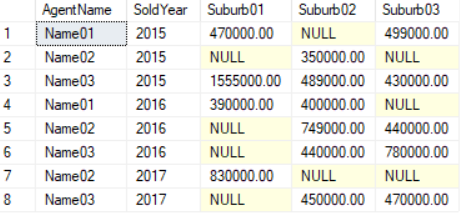
          FROM      Suburb3 s3

                    INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch



3.6. 6 columns in derived table

--================================================================

--T018\_03\_06

--HouseSoldRecord3 Table

--Pivot need derived table

--6 columns in derived table :

--AgentName, YEAR(SoldDateTime), DATEPART(MONTH, SoldDateTime), DATENAME(MM,SoldDateTime), SoldSuburb, SoldPrice

------------------------------------

--T018\_03\_06\_01

SELECT  \*

FROM    --derived table

        ( SELECT    hsr3.AgentName ,

                    YEAR(hsr3.SoldDateTime) AS SoldYear ,

                    DATENAME(MM, hsr3.SoldDateTime) AS SoldMonthName ,

                    s3.SuburbName AS SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      Suburb3 s3

                    INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

        ) AS BaseData PIVOT

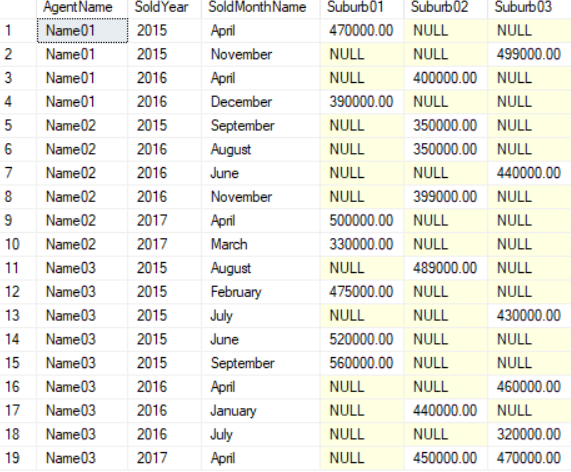
             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable

ORDER BY AgentName ,

        SoldYear ,

        SoldMonthName;

GO -- Run the previous command and begins new batch



------------------------------------

--T018\_03\_06\_02

SELECT  AgentName ,

        SoldYear ,

        SoldMonthName ,

        Suburb01 ,

        Suburb02 ,

        Suburb03

FROM    --derived table

        ( SELECT    hsr3.AgentName ,

                    YEAR(hsr3.SoldDateTime) AS SoldYear ,

                    DATEPART(MONTH, SoldDateTime) AS SoldMonth ,

                    DATENAME(MM, hsr3.SoldDateTime) AS SoldMonthName ,

                    s3.SuburbName AS SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      Suburb3 s3

                    INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

        ) AS BaseData PIVOT

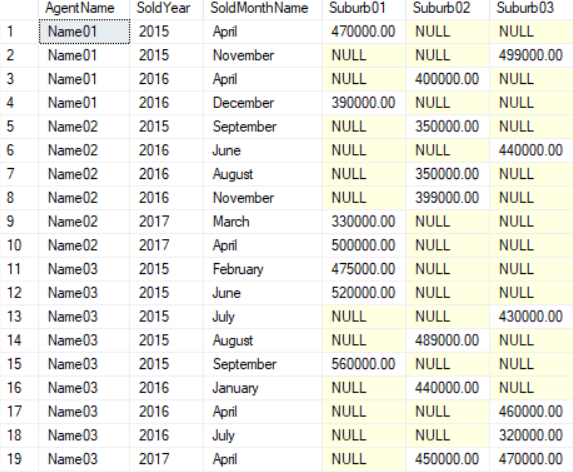
             ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable

ORDER BY AgentName ,

        SoldYear ,

        SoldMonth;

GO -- Run the previous command and begins new batch



/\*

1.

Pivot Syntax2

Reference:

[https://technet.microsoft.com/en-us/library/ms177410(v=sql.105).aspx](https://technet.microsoft.com/en-us/library/ms177410%28v=sql.105%29.aspx)

If you don't want to display all pivoted columns in left hand side.

Then you cannot use "SELECT \*",

you have to use "SELECT T1C1, T1C2, ..." in outter query.

--SELECT  T1C1, T1C2, ...T1Cn-3, T1Cn-1V1, T1Cn-1V2, T1Cn-1V3

---- You hide T1Cn-2 pivoted column, but You still can ORDER BY T1Cn-2.

--FROM    --derived table

--        ( SELECT    T1C1 , --1st pivoted column

--                    T1C2 , --2nd pivoted column

--                    ....

--                    T1Cn-3 , --N-3 th pivoted column

--                    T1Cn-2 , --N-2 th pivoted column which you don't want to display.

--                    T1Cn-1 , --Column n-1 that contains the values that will become column headers

--                    T1Cn AS T1CnAliasName --Column n that used for aggregation function

--          FROM      T1

--      ) AS BaseData PIVOT

--      ( SUM(T1CnAliasName) FOR T1Cn-2 IN ( T1Cn-1V1, T1Cn-1V2, T1Cn-1V3 ) ) AS PivotTable;

--ORDER BY T1C1, T1C2, ... , T1Cn-2

--GO -- Run the previous command and begins new batch

T stand for Table

C stand for column

V stand for Value

T1C1, T1C2, ... , T1Cn-3 will become the pivoted columns in left hand side.

You hide T1Cn-2 pivoted column, but You still can ORDER BY T1Cn-2.

Column n-1,T1Cn-1, that contains the values that will become column headers.

Column n,T1Cn, that used for aggregation function.

Using "ORDER BY T1Cn-1V1, T1Cn-1V2, T1Cn-1V3, T1Cn" might cause some logic error that we don't expect.

Better just use "ORDER BY T1C1, T1C2, ... , T1Cn-2".

---------------------------

1.1.

E.g.

--SELECT  AgentName ,

--        SoldYear ,

--        SoldMonthName ,

--        Suburb01 ,

--        Suburb02 ,

--        Suburb03

--FROM    --derived table

--        ( SELECT    hsr3.AgentName ,

--                    YEAR(hsr3.SoldDateTime) AS SoldYear ,

--                               DATEPART(MONTH, SoldDateTime) AS SoldMonth ,

--                    DATENAME(MM, hsr3.SoldDateTime) AS SoldMonthName ,

--                    s3.SuburbName AS SoldSuburb ,

--                    SoldPrice AS TotalSales

--          FROM      Suburb3 s3

--                    INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

--        ) AS BaseData PIVOT

--           ( SUM(TotalSales) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable

--ORDER BY AgentName ,

--        SoldYear ,

--        SoldMonth;

--GO -- Run the previous command and begins new batch

1.1.1.

AgentName, SoldYear, SoldMonthName will become the pivoted columns in left hand side.

Column SoldSuburb that contains the values that will become column headers, Suburb01, Suburb02, Suburb03.

Column SoldPrice that used for aggregation function.

Using "ORDER BY Suburb01, Suburb02, Suburb03, SoldSuburb" might cause some logic error that we don't expect.

Better just use "AgentName, SoldYear, SoldMonth".

1.1.2.

If using "ORDER BY AgentName , SoldYear, SoldMonthName"

SoldMonthName will become the alphabet order.

E.g. "April", "December", "July", "June" ,"November" ...etc.

This is not what we want,

we want the order by SoldMonth number but display SoldMonthName

E.g. "April", ... , "June", "July",..., "November" ,"December"

Thus, inner derived table SELECT both  SoldMonth and SoldMonthName.

But outer pivot table only SELECT SoldMonthName.

In addition, we still can ORDER BY SoldMonth.

\*/

3.7. dynamic sql query

--================================================================

--T018\_03\_07

--HouseSoldRecord3 Table

--dynamic sql query

3.7.1. fnGetAllSuburb

----------------------------------------------------------------------------

--T018\_03\_07\_01

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnGetAllSuburb' ) )

    BEGIN

        DROP FUNCTION fnGetAllSuburb;

    END;

GO -- Run the previous command and begins new batch

CREATE FUNCTION fnGetAllSuburb ( )

RETURNS NVARCHAR(MAX)

AS

    BEGIN

        DECLARE @AllSuburbName NVARCHAR(MAX) = '';

        SELECT  @AllSuburbName += ',' + QUOTENAME(SuburbName)

             -- QUOTENAME(SuburbName, '[]')

             -- QUOTENAME(SuburbName, '()')

             -- QUOTENAME(SuburbName, '''')

        FROM    dbo.Suburb3

        ORDER BY SuburbName;

             --E.g. ,[Suburb01],[Suburb02],[Suburb03]

             --Thus, need to get rid of first ','

        SET @AllSuburbName = SUBSTRING(@AllSuburbName, 2,

                                       LEN(@AllSuburbName) - 1);

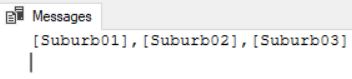
        RETURN  @AllSuburbName;

    END;

GO -- Run the previous command and begins new batch

PRINT dbo.fnGetAllSuburb()

GO -- Run the previous command and begins new batch



3.7.2. sp\_executesql

----------------------------------------------------------------------------

--T018\_03\_07\_02

DECLARE @AllSuburbName NVARCHAR(MAX) = dbo.fnGetAllSuburb();

PRINT @AllSuburbName;

DECLARE @Sql NVARCHAR(MAX) = '

SELECT  AgentName ,

        SoldYear ,

        SoldMonthName ,

        Suburb01 ,

        Suburb02 ,

        Suburb03

FROM    --derived table

        ( SELECT    hsr3.AgentName ,

                    YEAR(hsr3.SoldDateTime) AS SoldYear ,

                                 DATEPART(MONTH, SoldDateTime) AS SoldMonth ,

                    DATENAME(MM, hsr3.SoldDateTime) AS SoldMonthName ,

                    s3.SuburbName AS SoldSuburb ,

                    SoldPrice AS TotalSales

          FROM      Suburb3 s3

                    INNER JOIN HouseSoldRecord3 hsr3 ON s3.SuburbId = hsr3.HouseSoldSuburbID

        ) AS BaseData PIVOT

             ( SUM(TotalSales) FOR SoldSuburb IN (' + @AllSuburbName

    + ') ) AS PivotTable

ORDER BY AgentName ,

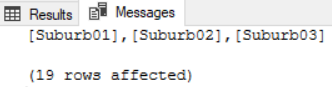
        SoldYear ,

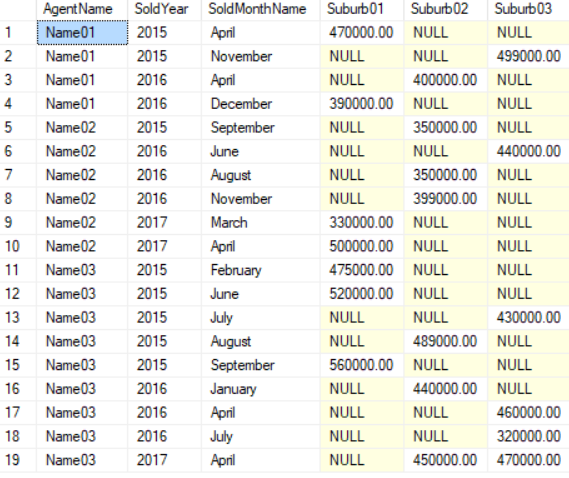
        SoldMonth;

';

EXEC sp\_executesql @Sql;

GO -- Run the previous command and begins new batch





==================================================

4. Clean up

--================================================================

--T018\_04\_Clean up

--================================================================

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord1' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord1;

        DROP TABLE HouseSoldRecord1;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord2' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord2;

        DROP TABLE HouseSoldRecord2;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord3' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord3;

        DROP TABLE HouseSoldRecord3;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'Suburb3' ) )

    BEGIN

        TRUNCATE TABLE dbo.Suburb3;

        DROP TABLE Suburb3;

    END;

GO -- Run the previous command and begins new batch

--If function exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'FUNCTION'

                        AND LEFT(ROUTINE\_NAME, 2) NOT IN ( '@@' )

                        AND SPECIFIC\_NAME = 'fnGetAllSuburb' ) )

    BEGIN

        DROP FUNCTION fnGetAllSuburb;

    END;

GO -- Run the previous command and begins new batch

5. PIVOT\_UNPIVOT : HouseSoldRecord4 Table

--================================================================

--T018\_05\_PIVOT\_UNPIVOT : HouseSoldRecord4 Table

--================================================================

5.1. Create Sample Data

--================================================================

--T018\_05\_01

--Create Sample Data

--There is no duplicate combination data of AgentName and SoldSuburb.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord4' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord4;

        DROP TABLE HouseSoldRecord4;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSoldRecord4

    (

      AgentName NVARCHAR(100) ,

      SoldSuburb NVARCHAR(100) ,

      SoldPrice MONEY

    );

GO -- Run the previous command and begins new batch

INSERT  HouseSoldRecord4

VALUES  ( N'Name01', N'Suburb02', 450000 );

INSERT  HouseSoldRecord4

VALUES  ( N'Name02', N'Suburb01', 475000 );

INSERT  HouseSoldRecord4

VALUES  ( N'Name02', N'Suburb02', 489000 );

INSERT  HouseSoldRecord4

VALUES  ( N'Name02', N'Suburb03', 399000 );

INSERT  HouseSoldRecord4

VALUES  ( N'Name01', N'Suburb03', 499000 );

INSERT  HouseSoldRecord4

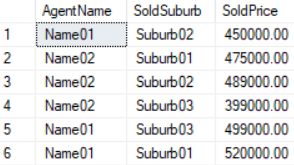
VALUES  ( N'Name01', N'Suburb01', 520000 );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    HouseSoldRecord4;

GO -- Run the previous command and begins new batch



5.2. Pivot need derived table

--================================================================

--T018\_05\_02

--HouseSoldRecord4 Table

--vwHouseSoldRecord4Pivot1

--Pivot need derived table

--Delete View if exist

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'vwHouseSoldRecord4Pivot1' ) )

    BEGIN

        DROP VIEW vwHouseSoldRecord4Pivot1;

    END;

GO -- Run the previous command and begins new batch

--Create view for HouseSoldRecord4 Povit Table

CREATE VIEW vwHouseSoldRecord4Pivot1

AS

    SELECT  AgentName ,

            Suburb01 ,

            Suburb02 ,

            Suburb03

    FROM    HouseSoldRecord4 PIVOT

( SUM(SoldPrice) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

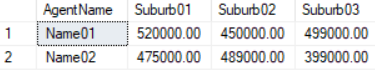
GO -- Run the previous command and begins new batch

--See the View data

SELECT  \*

FROM    vwHouseSoldRecord4Pivot1;

GO -- Run the previous command and begins new batch



/\*

1.

Output as the following

--AgentName Suburb01   Suburb02   Suburb03

--Name01    520000.00  450000.00  499000.00

--Name02    475000.00  489000.00  399000.00

2.

Pivot need derived table

2.1.

The PIVOT query for HouseSoldRecord4 converts the unique column values (Suburb01, Suburb02, Suburb03)

in SoldSuburb column into Columns in the output,

along with performing aggregations on the SoldPrice column.

The Outer query, simply, selects AgentName column from HouseSoldRecord4 table,

along with pivoted columns from the PivotTable.

2.2.

In real world, Table should have any number of columns.

However,

HouseSoldRecord4 only has 3 columns, AgentName, SoldSuburb, and SoldPrice.

Not every table only has 3 columns.

There will be a 'logic error' if the table has more than 3 columns.

2.3.

In HouseSoldRecord4,

there is no duplicate combination data of AgentName and SoldSuburb.

Thus, SUM(SoldPrice) aggregations is actually not doing anything.

Hense, This vwHouseSoldRecord4Pivot1 is ok to UNPIVOT.

\*/

5.3. The following clauses are equivalent

--================================================================

--T018\_05\_03

--The following clauses are equivalent

----------------------------------------------------------------------------

--T018\_05\_03\_01

--UNPIVOT vwHouseSoldRecord4Pivot1

SELECT AgentName, SoldSuburb, SoldPrice

FROM vwHouseSoldRecord4Pivot1

UNPIVOT

(

       SoldPrice

       FOR SoldSuburb IN (Suburb01, Suburb02 ,Suburb03)

) AS UnpivotExample

ORDER BY AgentName, SoldSuburb;

GO -- Run the previous command and begins new batch

----------------------------------------------------------------------------

--T018\_05\_03\_02

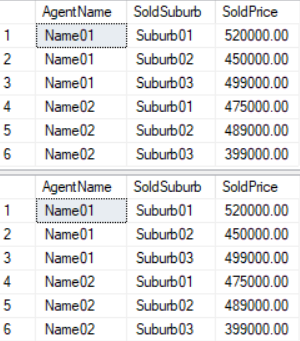
--The orginal HouseSoldRecord4

SELECT  \*

FROM    HouseSoldRecord4

ORDER BY AgentName, SoldSuburb;

GO -- Run the previous command and begins new batch



==================================================

6. PIVOT\_UNPIVOT : HouseSoldRecord5 Table

--================================================================

--T018\_06\_PIVOT\_UNPIVOT : HouseSoldRecord5 Table

--================================================================

6.1. Create Sample Data

--================================================================

--T018\_06\_01

--Create Sample Data

--There are some duplicate combination data of AgentName and SoldSuburb.

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord5' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord5;

        DROP TABLE HouseSoldRecord5;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE HouseSoldRecord5

    (

      AgentName NVARCHAR(100) ,

      SoldSuburb NVARCHAR(100) ,

      SoldPrice MONEY

    );

GO -- Run the previous command and begins new batch

INSERT  HouseSoldRecord5

VALUES  ( N'Name01', N'Suburb02', 450000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name02', N'Suburb01', 475000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name02', N'Suburb02', 489000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name02', N'Suburb03', 399000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name01', N'Suburb03', 499000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name01', N'Suburb01', 520000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name01', N'Suburb02', 345000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name02', N'Suburb01', 445000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name02', N'Suburb02', 555000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name02', N'Suburb03', 665000 );

INSERT  HouseSoldRecord5

VALUES  ( N'Name01', N'Suburb03', 477000 );

INSERT  HouseSoldRecord5

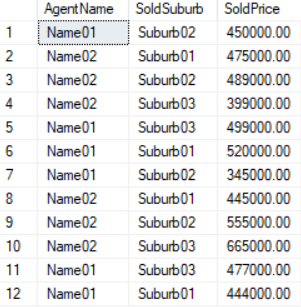
VALUES  ( N'Name01', N'Suburb01', 444000 );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    HouseSoldRecord5;

GO -- Run the previous command and begins new batch



6.2. Pivot need derived table

--================================================================

--T018\_06\_02

--HouseSoldRecord5 Table

--vwHouseSoldRecord5Pivot1

--Pivot need derived table

--Delete View if exist

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'vwHouseSoldRecord5Pivot1' ) )

    BEGIN

        DROP VIEW vwHouseSoldRecord5Pivot1;

    END;

GO -- Run the previous command and begins new batch

--Create view for HouseSoldRecord5 Povit Table

CREATE VIEW vwHouseSoldRecord5Pivot1

AS

    SELECT  AgentName ,

            Suburb01 ,

            Suburb02 ,

            Suburb03

    FROM    HouseSoldRecord5 PIVOT

( SUM(SoldPrice) FOR SoldSuburb IN ( Suburb01, Suburb02, Suburb03 ) ) AS PivotTable;

GO -- Run the previous command and begins new batch

--See the View data

SELECT  \*

FROM    vwHouseSoldRecord5Pivot1;

GO -- Run the previous command and begins new batch

A picture containing table

Description automatically generated

/\*

1.

Output as the following

--AgentName Suburb01   Suburb02   Suburb03

--Name01    964000.00  795000.00  976000.00

--Name02    920000.00  1044000.00 1064000.00

2.

Pivot need derived table

2.1.

The PIVOT query for HouseSoldRecord5 converts the unique column values (Suburb01, Suburb02, Suburb03)

in SoldSuburb column into Columns in the output,

along with performing aggregations on the SoldPrice column.

The Outer query, simply, selects AgentName column from HouseSoldRecord5 table,

along with pivoted columns from the PivotTable.

2.2.

In real world, Table should have any number of columns.

However,

HouseSoldRecord5 only has 3 columns, AgentName, SoldSuburb, and SoldPrice.

Not every table only has 3 columns.

There will be a 'logic error' if the table has more than 3 columns.

2.3.

In HouseSoldRecord5,

there are some duplicate combination data of AgentName and SoldSuburb.

Thus, SUM(SoldPrice) aggregations is actually doing anything.

Hense, This vwHouseSoldRecord5Pivot1 is NOT ok to UNPIVOT.

\*/

6.3. The following clauses are NOT equivalent

--================================================================

--T018\_06\_03

--The following clauses are NOT equivalent

--T018\_06\_03\_01

--UNPIVOT vwHouseSoldRecord5Pivot1

SELECT AgentName, SoldSuburb, SoldPrice

FROM vwHouseSoldRecord5Pivot1

UNPIVOT

(

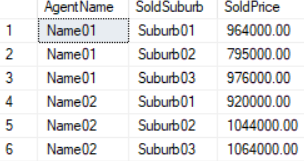
       SoldPrice

       FOR SoldSuburb IN (Suburb01, Suburb02 ,Suburb03)

) AS UnpivotExample

ORDER BY AgentName, SoldSuburb;

GO -- Run the previous command and begins new batch



-----------------------------------------------------

--T018\_06\_03\_02

--The orginal HouseSoldRecord5

SELECT  \*

FROM    HouseSoldRecord5

ORDER BY AgentName, SoldSuburb;

GO -- Run the previous command and begins new batch

/\*

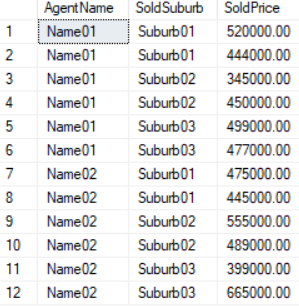
If the PIVOT operator has not aggregated the data,

you can get your original data back using the UNPIVOT operator

but If the PIVOT operator has aggregated the data,

then you can NOT use UNPIVOT operator.

\*/



==================================================

7. Clean up

--================================================================

--T018\_07\_Clean up

--================================================================

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord4' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord4;

        DROP TABLE HouseSoldRecord4;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'HouseSoldRecord5' ) )

    BEGIN

        TRUNCATE TABLE dbo.HouseSoldRecord5;

        DROP TABLE HouseSoldRecord5;

    END;

GO -- Run the previous command and begins new batch

--Delete View if exist

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'vwHouseSoldRecord4Pivot1' ) )

    BEGIN

        DROP VIEW vwHouseSoldRecord4Pivot1;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'vwHouseSoldRecord5Pivot1' ) )

    BEGIN

        DROP VIEW vwHouseSoldRecord5Pivot1;

    END;

GO -- Run the previous command and begins new batch