**Using PIVOT and UNPIVOT**

You can use the PIVOT and UNPIVOT relational operators to change a table-valued expression into another table. PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output, and runs aggregations where they're required on any left over column values that are wanted in the final output.

UNPIVOT carries out the opposite operation to PIVOT by rotating columns of a table-valued expression into column values.

Create an Employee Table

**CREATE** **TABLE** Employee

(

**Name** [nvarchar](**max**),

   [Year] [**int**] ,

   Sales [**int**]

)

Insert the following data into the table

**INSERT** **INTO** Employee

**SELECT** 'Pankaj',2010,72500 **UNION** ALL

**SELECT** 'Rahul',2010,60500 **UNION** ALL

**SELECT** 'Sandeep',2010,52000 **UNION** ALL

**SELECT** 'Pankaj',2011,45000 **UNION** ALL

**SELECT** 'Sandeep',2011,82500 **UNION** ALL

**SELECT** 'Rahul',2011,35600 **UNION** ALL

**SELECT** 'Pankaj',2012,32500 **UNION** ALL

**SELECT** 'Pankaj',2010,20500 **UNION** ALL

**SELECT** 'Rahul',2011,200500 **UNION** ALL

**SELECT** 'Sandeep',2010,32000

Now we check data of Employee table.

**SELECT** \* **FROM** Employee;

## PIVOT

PIVOT relational operator convert data from row level to column level. PIVOT rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output. Using PIVOT operator we can perform aggregate operation where we required.  
  
**Syntax**

**SELECT** <non-pivoted **column**>,

       <list **of** pivoted **column**>

**FROM**

(<**SELECT** query  **to** produces the data>)

**AS** <alias **name**>

PIVOT

(

<aggregation **function**>(<**column** **name**>)

**FOR**

[<**column** **name** that  become **column** headers>]

    IN ( [list **of**  pivoted columns])

) **AS** <alias **name**  **for**  pivot **table**>

**Example 1**

**SELECT** [Year], Pankaj,Rahul,Sandeep **FROM**

(**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1

PIVOT

(

SUM(Sales) **FOR** **Name** IN (Pankaj,Rahul,Sandeep)) **AS** Tab2

**ORDER** **BY** [Tab2].[Year]

In above query we calculate the sum of sales for Pankaj, Rahul and Sandeep employee corresponding to year value.  
  
**Example 2 : Error (col name cannot be number)**

**SELECT** **Name**, 2010,2011,2012 **FROM**

(**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1

PIVOT

(

SUM(Sales) **FOR** [Year] IN (2010,2011,2012)) **AS** Tab2

**ORDER** **BY** Tab2.**Name**

When we execute above query, SQL Server throws an error because we can’t provide integer value as a column name directly. To remove this error use the brackets before each integer value as in the following code snippet:

**SELECT** **Name**, [2010],[2011],[2012] **FROM**

(**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1

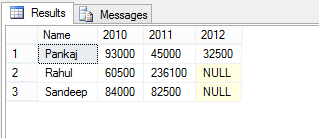
PIVOT

(

SUM(Sales) **FOR** [Year] IN ([2010],[2011],[2012])) **AS** Tab2

**ORDER** **BY** Tab2.**Name**

**Output**



## UNPIVOT in SQL Server

UNPIVOT relational operator is reverse process of PIVOT relational operator. UNPIVOT relational operator convert data from column level to row level.  
  
**Example 3**  
  
Suppose that output of example 2 is stored in Temp Variable. Now we want to rotate column identifiers Pankaj, Sandeep, Rahul into row values. For this we use the UNPIVOT relational operator.  
  
**Declare Temp Table**

**CREATE TABLE**  #Tab

(

   [Year] **int**,

   Pankaj **int**,

   Rahul **int**,

   Sandeep **int**

)

**Insert Value in Temp Variable**

**INSERT** **INTO** #Tab

**SELECT** [Year], Pankaj,Rahul,Sandeep **FROM**

(**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1

PIVOT

(

   SUM(Sales) **FOR** **Name** IN (Pankaj,Rahul,Sandeep)) **AS** Tab2

**ORDER** **BY** [Tab2].[Year]

**Perform UNPIVOT Operation**

**SELECT** **Name**,[Year] , Sales **FROM** @Tab t

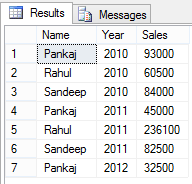
UNPIVOT

(

Sales **FOR** **Name** IN (Pankaj,Rahul,Sandeep)

) **AS** TAb2

**Output**



We can perform first PIVOT operation and after that UNPIVOT operation on same table in single query as in the following code snippet.

**SELECT** **Name**,[Year] , Sales **FROM**

(

**SELECT** [Year], Pankaj,Rahul,Sandeep **FROM**

   (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1

PIVOT

(

   SUM(Sales) **FOR** **Name** IN (Pankaj,Rahul,Sandeep)) **AS** Tab2

)Tab

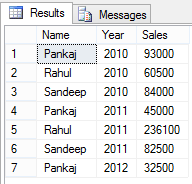
UNPIVOT

(

   Sales **FOR** **Name** IN (Pankaj,Rahul,Sandeep)

) **AS** TAb2

**Output**



**Note**  
UNPIVOT operation is a reverse process of PIVOT operation, but UNPIVOT is not the exact reverse of PIVOT. If PIVOT performs an aggregation and merges multiple rows into a single row in the output, then UNPIVOT can’t reproduce the original table-valued expression result because rows have been merged. So conclusion is that if PIVOT operation merges multiple row in a single row, then UNPIVOT operation can’t retrieve original table from the output of PIVOT operation. But if PIVOT operation doesn’t merge multiple row in a single row, then UNPIVOT operation can retrieve original table from the output of PIVOT operation.