PIVOT and UNPIVOT are two relational operators that are used to convert a table expression into another. PIVOT is used when we want to transfer data from row level to column level and UNPIVOT is used when we want to convert data from column level to row level. PIVOT and UNPIVOT relational operators are used to generate a multidimensional reporting. Today we will discuss both the operators. PIVOT and UNPIVOT relational operators are used to generate an interactive table that quickly combines and compares large amount of data.

Firstly, we will create a table and insert some data into the table.  
  
Firstly, create an Employee Table

1. **CREATE** **TABLE** Employee
2. (
3. **Name** [nvarchar](**max**),
4. [Year] [**int**] ,
5. Sales [**int**]
6. )

Insert the following data into the table

1. **INSERT** **INTO** Employee
2. **SELECT** 'Pankaj',2010,72500 **UNION** ALL
3. **SELECT** 'Rahul',2010,60500 **UNION** ALL
4. **SELECT** 'Sandeep',2010,52000 **UNION** ALL
5. **SELECT** 'Pankaj',2011,45000 **UNION** ALL
6. **SELECT** 'Sandeep',2011,82500 **UNION** ALL
7. **SELECT** 'Rahul',2011,35600 **UNION** ALL
8. **SELECT** 'Pankaj',2012,32500 **UNION** ALL
9. **SELECT** 'Pankaj',2010,20500 **UNION** ALL
10. **SELECT** 'Rahul',2011,200500 **UNION** ALL
11. **SELECT** 'Sandeep',2010,32000

Now we check data of Employee table.

1. **SELECT** \* **FROM** Employee;

**Output**  
  
  
  
We use above Employee table for PIVOT and UNPIVOT relational operator examples. First we read about PIVOT realational operator.  
  
**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**

1. **SELECT** <non-pivoted **column**>,
2. <list **of** pivoted **column**>
3. **FROM**
4. (<**SELECT** query  **to** produces the data>)
5. **AS** <alias **name**>
6. PIVOT
7. (
8. <aggregation **function**>(<**column** **name**>)
9. **FOR**
10. [<**column** **name** that  become **column** headers>]
11. IN ( [list **of**  pivoted columns])
13. ) **AS** <alias **name**  **for**  pivot **table**>

Let us take some examples.  
  
**Example 1**

1. **SELECT** [Year], Pankaj,Rahul,Sandeep **FROM**
2. (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1
3. PIVOT
4. (
5. SUM(Sales) **FOR** **Name** IN (Pankaj,Rahul,Sandeep)) **AS** Tab2
6. **ORDER** **BY** [Tab2].[Year]

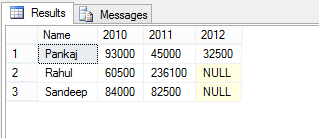
**Output:**

In above query we calculate the sum of sales for Pankaj, Rahul and Sandeep employee corresponding to year value.  
  
**Example 2**

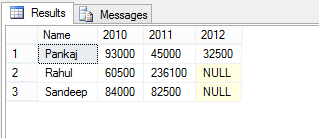
1. **SELECT** **Name**, 2010,2011,2012 **FROM**
2. (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1
3. PIVOT
4. (
5. SUM(Sales) **FOR** [Year] IN (2010,2011,2012)) **AS** Tab2
6. **ORDER** **BY** Tab2.**Name**

**Output:**  
  
  
  
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:

1. **SELECT** **Name**, [2010],[2011],[2012] **FROM**
2. (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1
3. PIVOT
4. (
5. SUM(Sales) **FOR** [Year] IN ([2010],[2011],[2012])) **AS** Tab2
6. **ORDER** **BY** Tab2.**Name**

**Output**  
  
  
  
**Example 3**  
In previous examples we wrote the name of pivot column. This approach is useful if we know all possible values for pivot column. But it is not fix that column always remain same, pivot column may be increased or decreased in future.  
  
Let us take previous example. In previous example we wrote 2010,2011 and 2012 as pivot column. But it is not fix that these column will not change in future , so what should we do if there is a possibility that column number may change in future.  
  
In such a condition we should use dynamic query. Firstly, retrieve all unique values from pivot column and after that write a dynamic query to execute it with pivot query at run time.  
  
Now we execute example 2, but using dynamic query.

1. /\***Declare** Variable\*/
2. **DECLARE** @Pivot\_Column [nvarchar](**max**);
3. **DECLARE** @Query [nvarchar](**max**);
5. /\***Select** Pivot **Column**\*/
6. **SELECT** @Pivot\_Column= COALESCE(@Pivot\_Column+',','')+ QUOTENAME(Year) **FROM**
7. (**SELECT** **DISTINCT** [Year] **FROM** Employee)Tab
9. /\***Create** **Dynamic** Query\*/
10. **SELECT** @Query='SELECT Name, '+@Pivot\_Column+'**FROM**
11. (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1
12. PIVOT
13. (
14. SUM(Sales) **FOR** [Year] IN ('+@Pivot\_Column+')) **AS** Tab2
15. **ORDER** **BY** Tab2.**Name**'
17. /\***Execute** Query\*/
18. **EXEC**  sp\_executesql  @Query

**Output**  
  
**UNPIVOT**  
UNPIVOT relational operator is reverse process of PIVOT relational operator. UNPIVOT relational operator convert data from column level to row level.  
  
**Example 4:**  
  
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 Variable**

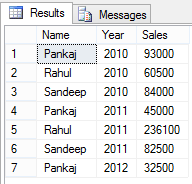
1. **DECLARE** @Tab **TABLE**
2. (
3. [Year] **int**,
4. Pankaj **int**,
5. Rahul **int**,
6. Sandeep **int**
7. )

**Insert Value in Temp Variable**

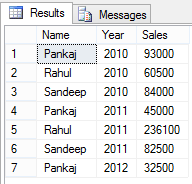
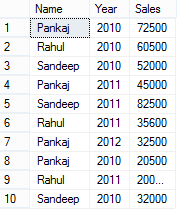
1. **INSERT** **INTO** @Tab
2. **SELECT** [Year], Pankaj,Rahul,Sandeep **FROM**
3. (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1
4. PIVOT
5. (
6. SUM(Sales) **FOR** **Name** IN (Pankaj,Rahul,Sandeep)) **AS** Tab2
7. **ORDER** **BY** [Tab2].[Year]

**Perform UNPIVOT Operation**

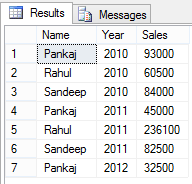
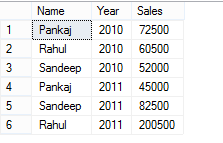
1. **SELECT** **Name**,[Year] , Sales **FROM** @Tab t
2. UNPIVOT
3. (
4. Sales **FOR** **Name** IN (Pankaj,Rahul,Sandeep)
5. ) **AS** TAb2

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

1. **SELECT** **Name**,[Year] , Sales **FROM**
2. (
3. **SELECT** [Year], Pankaj,Rahul,Sandeep **FROM**
4. (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1
5. PIVOT
6. (
7. SUM(Sales) **FOR** **Name** IN (Pankaj,Rahul,Sandeep)) **AS** Tab2
8. )Tab
9. UNPIVOT
10. (
11. Sales **FOR** **Name** IN (Pankaj,Rahul,Sandeep)
12. ) **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.  
  
**Let us take an example**  
  
**Case 1(PIVOT Merger Multiple Row) :**  
  
Now we perform PIVOT and UNPIVOT operation for this table and compare the resultant table from this table.

1. **SELECT** **Name**,[Year] , Sales **FROM**
2. (
3. **SELECT** [Year], Pankaj,Rahul,Sandeep **FROM**
4. (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1
5. PIVOT
6. (
7. SUM(Sales) **FOR** **Name** IN (Pankaj,Rahul,Sandeep)) **AS** Tab2
8. )Tab
9. UNPIVOT
10. (
11. Sales **FOR** **Name** IN (Pankaj,Rahul,Sandeep)
12. ) **AS** TAb2

**Output**  
  
  
  
We can see that both the tables are not same. First table contain 10 rows but the above  table contains only 7 rows. This difference occur due to the PIVOT operation. PIVOT operation merge the (4,7,8) row in a single row and merge the (3,5) row in a single row. So UNPIVOT operation can’t retrieve original table.  
  
**Case 2(PIVOT doesn’t Merger Multiple Row)**  
  
Now we perform PIVOT and UNPIVOT operation for this table and compare the resultant table from this table.

1. **SELECT** **Name**,[Year] , Sales **FROM**
2. (
3. **SELECT** [Year], Pankaj,Rahul,Sandeep **FROM**
4. (**SELECT** **Name**, [Year] , Sales **FROM** Employee )Tab1
5. PIVOT
6. (
7. SUM(Sales) **FOR** **Name** IN (Pankaj,Rahul,Sandeep)) **AS** Tab2
8. )Tab
9. UNPIVOT
10. (
11. Sales **FOR** **Name** IN (Pankaj,Rahul,Sandeep)
12. ) **AS** TAb2

**Output**