

Module 14

Pivoting and Grouping Sets



Zyead Ahmed.
Aspiring To Learning Data Engineer.

Module Overview

- Writing Queries with PIVOT and UNPIVOT
- Working with Grouping Sets

Lesson 1: Writing Queries with PIVOT and UNPIVOT

- What Is Pivoting?
- Elements of PIVOT
- Writing Queries with UNPIVOT
- Demonstration: Writing Queries with PIVOT and UNPIVOT

What Is Pivoting?

Pivoting data is rotating data from a rows-based orientation to a columns-based orientation

Distinct values from a single column are projected across as headings for other columns—may include aggregation

Category	Qty	Orderyear
Dairy Products	12	2006
Grains/Cereals	10	2006
Dairy Products	5	2006
Produce	9	2006
Produce	40	2006
Seafood	10	2006
Produce	35	2006
Condiments	15	2006
Grains/Cereals	6	2006
Grains/Cereals	15	2006
Condiments	20	2006
Confections	40	2006
Dairy Products	25	2006
Dairy Products	40	2006
Dairy Products	20	2006

Category	2006	2007	2008
Beverages	1842	3996	3694
Condiments	962	2895	1441
Confections	1357	4137	2412
Dairy Products	2086	4374	2689
Grains/Cereals	549	2636	1377
Meat/Poultry	950	2189	1060
Produce	549	1583	858
Seafood	1286	3679	2716



Pivoted data

Elements of PIVOT

- Pivoting includes three phases:
 1. Grouping determines which element gets a row in the result set
 2. Spreading provides the distinct values to be pivoted across
 3. Aggregation performs an aggregation function (such as SUM)

Elements of PIVOT

```
SELECT Category, [2006],[2007],[2008]
FROM (SELECT Category, Qty, Orderyear
      FROM Sales.CategoryQtyYear) AS D
PIVOT(SUM(QTY) FOR orderyear
      IN ([2006],[2007],[2008])) AS pvt
ORDER BY Category;
```

Category	2006	2007	2008
-----	----	----	----
Beverages	1842	3996	3694
Condiments	962	2895	1441
Confections	1357	4137	2412
Dairy Products	2086	4374	2689
Grains/Cereals	549	2636	1377
Meat/Poultry	950	2189	1060
Produce	549	1583	858
Seafood	1286	3679	2716

Writing Queries with UNPIVOT

- Unpivoting data is rotating data from a columns-based orientation to a rows-based orientation
- Spreads or splits values from one source row into one or more target rows
- Each source row becomes one or more rows in result set based on number of columns being pivoted
- Unpivoting includes three elements:
 - Source columns to be unpivoted
 - Name to be assigned to new values column
 - Name to be assigned to names columns

Demonstration: Writing Queries with PIVOT and UNPIVOT

In this demonstration, you will see how to:

- Use PIVOT and UNPIVOT

Lesson 2: Working with Grouping Sets

- Writing Queries with Grouping Sets
- CUBE and ROLLUP
- GROUPING_ID
- Demonstration: Using Grouping Sets

Writing Queries with Grouping Sets

- GROUPING SETS subclause builds on T-SQL GROUP BY clause
- Allows multiple groupings to be defined in same query
- Alternative to use of UNION ALL to combine multiple outputs (each with different GROUP BY) into one result set

```
SELECT <column list with aggregate(s)>
FROM <source>
GROUP BY
GROUPING SETS(
    (<column_name>), --one or more columns
    (<column_name>), --one or more columns
    () -- empty parentheses if aggregating all rows
);
```

Writing Queries with Grouping Sets

```
SELECT Category, Cust, SUM(Qty) AS TotalQty
FROM Sales.CategorySales
GROUP BY
GROUPING SETS((Category),(Cust),());
```

Category	Cust	TotalQty
-----	-----	-----
NULL	NULL	999
NULL	1	80
NULL	2	12
NULL	3	154
NULL	4	241
NULL	5	512
Beverages	NULL	513
Condiments	NULL	114
Confections	NULL	372

CUBE and ROLLUP

- CUBE provides shortcut for defining grouping sets given a list of columns
- All possible combinations of grouping sets created

```
SELECT Category, Cust, SUM(Qty) AS TotalQty  
FROM Sales.CategorySales  
GROUP BY CUBE(Category,Cust)  
ORDER BY Category, Cust;
```

- ROLLUP provides shortcut for defining grouping sets, creates combinations assuming input columns form a hierarchy

```
SELECT Category, Cust, SUM(Qty) AS TotalQty  
FROM Sales.CategorySales  
GROUP BY ROLLUP(Category,Cust)  
ORDER BY Category, Cust;
```

GROUPING_ID

- Multiple grouping sets present a problem in identifying the source of each row in the result set
- NULLs could come from the source data or could be a placeholder in the grouping set
- The GROUPING_ID function provides a method to mark a row with a 1 or 0 to identify which grouping set the row is a member of

```
SELECT GROUPING_ID(Category) AS grpCat,  
       GROUPING_ID(Cust) AS grpCust,  
       Category, Cust, SUM(Qty) AS TotalQty  
FROM Sales.CategorySales  
GROUP BY CUBE(Category, Cust)  
ORDER BY Category, Cust;
```

Demonstration: Using Grouping Sets

In this demonstration, you will see how to:

- Use the CUBE and ROLLUP subclauses