Lab Answer Key: Module 14: Pivoting and Grouping Sets

Lab: Pivoting and Grouping Sets

Exercise 1: Writing Queries That Use the PIVOT Operator

Task 1: Prepare the Lab Environment

1. Ensure that the **20761C-MIA-DC** and **20761C-MIA-SQL** virtual machines are both running, and then log on to **20761C-MIA-SQL** as **ADVENTUREWORKS\Student** with the password **Pa55w.rd**.
2. In the **D:\Labfiles\Lab14\Starter** folder, right-click **Setup.cmd**, and then click **Run as administrator**.
3. In the **User Account Control** dialog box, click **Yes**, and then wait for the script to finish.

Task 2: Write a SELECT Statement to Retrieve the Number of Customers for a Specific Customer Group

1. Start SQL Server Management Studio and connect to the **MIA-SQL** database engine using Windows authentication.
2. On the **File** menu, click **Open** and click **Project/Solution**.
3. In the **Open Project** window, open the project **D:\Labfiles\Lab14\Starter\Project\Project.ssmssln**.
4. In Solution Explorer, double-click the query **51 - Lab Exercise 1.sql**.
5. In the query window, highlight the statement **USE TSQL;** and click **Execute**.
6. Highlight the following provided T-SQL code:

```
CREATE VIEW Sales.CustGroups AS
SELECT
custid,
CHOOSE(custid % 3 + 1, N'A', N'B', N'C') AS custgroup,
Country
FROM Sales.Customers;
```

7. Click **Execute**. This code creates a view named Sales.CustGroups.
8. In the query pane, type the following query after the provided T-SQL code:

```
SELECT
custid,
custgroup,
```

```
country
FROM Sales.CustGroups;
```

9. Highlight the written query and click **Execute**.
10. Modify the written T-SQL code by applying the PIVOT operator. The query should look like this:

```
SELECT
country,
p.A,
p.B,
p.C
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

11. Highlight the written query and click **Execute**.

Task 3: Specify the Grouping Element for the PIVOT Operator

1. Highlight the following provided T-SQL code after the **Task 2** description:

```
ALTER VIEW Sales.CustGroups AS
SELECT
custid,
CHOOSE(custid % 3 + 1, N'A', N'B', N'C') AS custgroup,
country,
city,
contactname
FROM Sales.Customers;
```

2. Click **Execute**. This code modifies the view by adding two additional columns.
3. Highlight the last query in task 1. On the toolbar, click **Edit** and then **Copy**.
4. In the query window, click the line after the provided T-SQL code. On the toolbar, click **Edit** and then **Paste**. The query should look like this:

```
SELECT
country,
p.A,
p.B,
p.C
```

```
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

5. Highlight the copied query and click **Execute**.
6. Observe the result. Is this result the same as that from the query in task 1? The result is not the same. More rows were returned after you modified the view.
7. Modify the copied T-SQL statement to include additional columns from the view. The query should look like this:

```
SELECT
country,
city,
contactname,
p.A,
p.B,
p.C
FROM Sales.CustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

8. Highlight the written query and click **Execute**.

Notice that you received the same result as the previous query. Why did you get the same number of rows? The PIVOT operator assumes that all the columns except the aggregation and spreading elements are part of the grouping columns.

Task 4: Use a Common Table Expression (CTE) to Specify the Grouping Element for the PIVOT Operator

1. In the query pane, type the following query after the **Task 3** description:

```
WITH PivotCustGroups AS
(
SELECT
custid,
country,
custgroup
FROM Sales.CustGroups
)
SELECT
country,
p.A,
p.B,
```

```
p.C
FROM PivotCustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

2. Highlight the written query and click **Execute**.

3. Observe the result. Is it the same as the result of the last query in task 1? Can you explain why? The result is the same. In this task, the CTE has provided three possible columns to the PIVOT operator. In task 1, the view also provided three columns to the PIVOT operator.

4. Why do you think it is beneficial to use a CTE when using the PIVOT operator? When using the PIVOT operator, you cannot directly specify the grouping element because SQL Server automatically assumes that all columns should be used as grouping elements, with the exception of the spreading and aggregation elements. With a CTE, you can specify the exact columns and therefore control that columns use for the grouping.

Task 5: Write a SELECT Statement to Retrieve the Total Sales Amount for Each Customer and Product Category

1. In the query pane, type the following query after the **Task 4** description:

```
WITH SalesByCategory AS
(
SELECT
o.custid,
d.qty * d.unitprice AS salesvalue,
c.categoryname
FROM Sales.Orders AS o
INNER JOIN Sales.OrderDetails AS d ON o.orderid = d.orderid
INNER JOIN Production.Products AS p ON p.productid = d.productid
INNER JOIN Production.Categories AS c ON c.categoryid = p.categoryid
WHERE o.orderdate >= '20080101' AND o.orderdate < '20090101'
)
SELECT
custid,
p.Beverages,
p.Condiments,
p.Confections,
p.[Dairy Products],
p.[Grains/Cereals],
p.[Meat/Poultry],
p.Produce,
p.Seafood
FROM SalesByCategory
PIVOT (SUM(salesvalue) FOR categoryname
```

```
IN (Beverages, Condiments, Confections, [Dairy Products], [Grains/Cereals],
[Meat/Poultry], Produce, Seafood)) AS p;
```

2. Highlight the written query and click **Execute**.

Result: After this exercise, you should be able to use the PIVOT operator in T-SQL statements.

Exercise 2: Writing Queries That Use the UNPIVOT Operator

Task 1: Create and Query the Sales.PivotCustGroups View

1. In Solution Explorer, double-click the query **61 - Lab Exercise 2.sql**.
2. In the query window, highlight the statement **USE TSQL;** and click **Execute**.
3. Highlight the following provided T-SQL code:

```
CREATE VIEW Sales.PivotCustGroups AS
WITH PivotCustGroups AS
(
SELECT
custid,
country,
custgroup
FROM Sales.CustGroups
)
SELECT
country,
p.A,
p.B,
p.C
FROM PivotCustGroups
PIVOT (COUNT(custid) FOR custgroup IN (A, B, C)) AS p;
```

4. Click **Execute**. This code creates a view named Sales.PivotCustGroups.
5. In the query pane, type the following query after the provided T-SQL code:

```
SELECT
country, A, B, C
FROM Sales.PivotCustGroups;
```

6. Highlight the written query and click **Execute**.

Task 2: Write a SELECT Statement to Retrieve a Row for Each Country and Customer Group

1. In the query pane, type the following query after the **Task 2** descriptions:

```
SELECT
custgroup,
country,
numberofcustomers
FROM Sales.PivotCustGroups
UNPIVOT (numberofcustomers FOR custgroup IN (A, B, C)) AS p;
```

2. Highlight the written query and click **Execute**.

Task 3: Remove the Created Views

- Highlight the provided T-SQL statement after the **Task 3** description and click **Execute**.

Result: After this exercise, you should know how to use the UNPIVOT operator in your T-SQL statements.

Exercise 3: Writing Queries That Use the GROUPING SETS, CUBE, and ROLLUP Subclauses

Task 1: Write a SELECT Statement That Uses the GROUPING SETS Subclause to Return the Number of Customers for Different Grouping Sets

1. In Solution Explorer, double-click the query **71 - Lab Exercise 3.sql**.
2. In the query window, highlight the statement **USE TSQL;** and click **Execute**.
3. In the query pane, type the following query after the **Task 1** description:

```
SELECT
country,
city,
COUNT(custid) AS noofcustomers
FROM Sales.Customers
GROUP BY
GROUPING SETS
```

```
(
(country, city),
(country),
(city),
()
);
```

4. Highlight the written query and click **Execute**.

Task 2: Write a SELECT Statement That Uses the CUBE Subclause to Retrieve Grouping Sets Based on Yearly, Monthly, and Daily Sales Values

1. In the query pane, type the following query after the **Task 2** description:

```
SELECT
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
DAY(orderdate) AS orderday,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
CUBE (YEAR(orderdate), MONTH(orderdate), DAY(orderdate));
```

2. Highlight the written query and click **Execute**.

Task 3: Write the Same SELECT Statement Using the ROLLUP Subclause

1. In the query pane, type the following query after the **Task 3** description:

```
SELECT
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
DAY(orderdate) AS orderday,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
ROLLUP (YEAR(orderdate), MONTH(orderdate), DAY(orderdate));
```

2. Highlight the written query and click **Execute**.

3. Observe the result. What is the difference between the ROLLUP and CUBE subclauses of the GROUP BY clause? Like the CUBE subclause, the ROLLUP subclause provides an abbreviated way to define multiple grouping sets. However, unlike CUBE, ROLLUP doesn't produce all possible grouping sets that can be defined based on the input members—it produces a subset of those. ROLLUP assumes a hierarchy among the input members and produces all grouping sets that make sense, considering the hierarchy. In other words, while CUBE(a, b, c) produces all eight possible grouping sets out of the three input members, ROLLUP(a, b, c) produces only four grouping sets, assuming the hierarchy a>b>c. ROLLUP(a, b, c) is the equivalent of specifying GROUPING SETS((a, b, c), (a, b), (a), ()).

Which is the more appropriate subclause to use in this example? Since year, month, and day form a hierarchy, the ROLLUP clause is more suitable. There is probably not much interest in showing aggregates for a month irrespective of year, but the other way around is interesting.

Task 4: Analyze the Total Sales Value by Year and Month

1. In the query pane, type the following query after the **Task 4** description:

```
SELECT
GROUPING_ID(YEAR(orderdate), MONTH(orderdate)) as groupid,
YEAR(orderdate) AS orderyear,
MONTH(orderdate) AS ordermonth,
SUM(val) AS salesvalue
FROM Sales.OrderValues
GROUP BY
ROLLUP (YEAR(orderdate), MONTH(orderdate))
ORDER BY groupid, orderyear, ordermonth;
```

2. Highlight the written query and click **Execute**.

3. Close SQL Server Management Studio without saving any changes.

Result: After this exercise, you should have an understanding of how to use the GROUPING SETS, CUBE, and ROLLUP subclauses in T-SQL statements.