For the next few questions, we will create a PIVOT to better illustrate 'Sales Total by Customers per Employee.'

 Write a Derived Table called dtPivot that includes the CustomerID, EmployeeID, and UnitPrice columns. You will need this for question #3. To solve this problem, you will need to do a join between the SALE and SALE\_ITEM Tables.

```
--1

⇒SELECT CustomerID, EmployeeID, UnitPrice
Into dtPivot
FROM (SELECT CustomerID, EmployeeID, UnitPrice
FROM Sale as S JOIN SALE_ITEM as I
ON S.SaleID = I.SaleID) as DT;

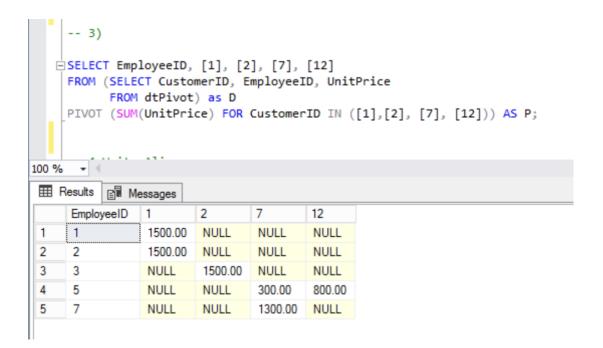
⇒Select *
FROM dtPivot
```

## Output:

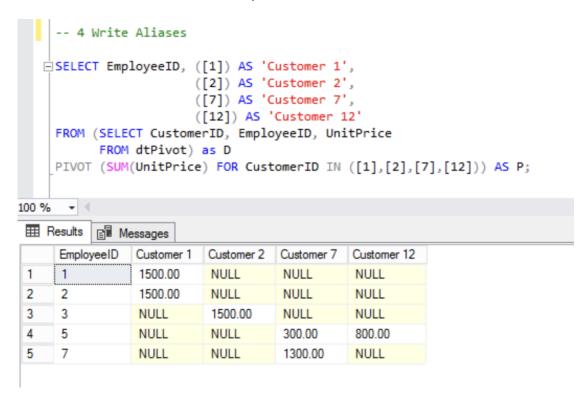
|   |            | ₽ Messages |            |           |
|---|------------|------------|------------|-----------|
|   | CustomerID |            | EmployeeID | UnitPrice |
| 1 | 1          |            | 1          | 1500.00   |
| 2 | 1          |            | 2          | 1500.00   |
| 3 | 2          |            | 3          | 1500.00   |
| 4 | 12         |            | 5          | 500.00    |
| 5 | 12         |            | 5          | 300.00    |
| 6 | 7          |            | 5          | 300.00    |
| 7 | 7          |            | 7          | 1300.00   |
|   |            |            |            |           |

2. In the Ben-Gan examples on Pivot, the Customer Column values consisted of letters (i.e., A, B, C). As you know, it's far more common to have integer values instead (i.e., 1, 2). The Pivot Operator requires you to put integer values in brackets (i.e., [1], [2]). This will prove useful as you complete this lab. To answer this question, write the following to prepare for question 3: **FOR CustomerID IN([1], [2])** 

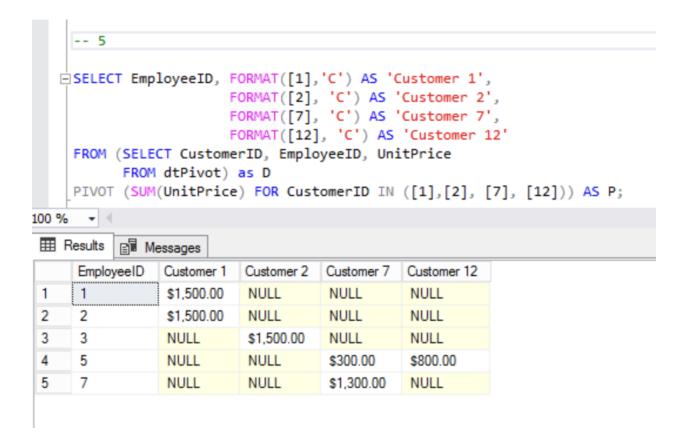
3. Use the Pivot Operator (refer to slides or text) to code unit price by customer per employeeID:



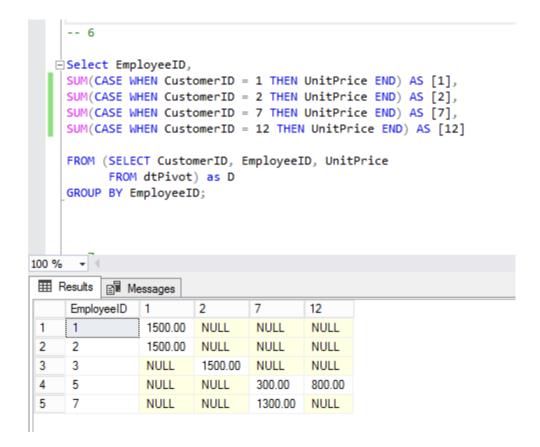
4. Rewrite the code above and place aliases in the SELECT clause.



5. One last improvement, for the above code, FORMAT the SELECT clause to change unit price totals into currency format.



6. In Question 3, you used the PIVOT Operator to solve the problem. This time, solve the same problem with Standard SQL (page 224 of text - Using a GROUP BY) to code UnitPrice totals by customer per employeeID:



7. Last thing, Add appropriate aliases and formatting to resemble the output you see below.

```
-- 7
   FORMAT(SUM(CASE When CustomerID = 1 THEN UnitPrice END), 'c') as [Customer1],
    FORMAT(SUM(CASE When CustomerID = 2 THEN UnitPrice END), 'c') as [Customer2],
    FORMAT(SUM(CASE When CustomerID = 5 THEN UnitPrice END), 'c') as [Customer7],
    FORMAT(SUM(CASE When CustomerID = 7 THEN UnitPrice END),'c') as [Customer7]
    FROM (SELECT CustomerID, EmployeeID, UnitPrice
         FROM dtPivot) as D
    GROUP BY EmployeeID;
100 % - <
Results Messages
     EmployeeID
               Customer1
                                          Customer7
                        Customer2 Customer7
     1
               $1,500.00
                       NULL
                                 NULL
                                          NULL
 1
 2
    2
               $1,500.00 NULL
                                 NULL
                                           NULL
 3
     3
               NULL
                        $1,500.00 NULL
                                          NULL
 4
     5
               NULL
                        NULL
                                 NULL
                                          $300.00
 5
     7
               NULL
                        NULL
                                 NULL
                                          $1,300.00
```