Coder Girl T-SQL Problem Set 6

There are many ways to use multiple tables in a query. This problem set covers subqueries, common table expressions, and UNION. Subqueries can be used for many things, such as an IN list, or to separate the logic of part of the query. Common table expressions (CTE) can be used to separate out part of the logic, but one CTE can be based on another CTE. UNION can be used to combine the results of two queries.

Watch the video on CTEs and subqueries to prepare for this problem set.

Exercise 1

1.1 Write a query that returns a list of the SalesOrderID, LineTotal, and ProductID for every order detail. Include a total for all sales for each product.

1.2 Write a query that returns the SalesOrderID, OrderDate, TotalDue, and CustomerID for each sale. Include the average TotalDue for each customer.

1.3 Modify the query written in 1.2 so that the difference between the total due and the average sale is also returned.

Exercise 2

2.1 The Production.Product table joins to the Production.ProductSubCategory table. That table joins to the Production.ProductCategory table. Write a query that joins these three tables. Return the ProductID plus the names of the product, subcategory and category. Remember to add aliases. This query is a regular join; you will use this query in subsequent steps.

2.2 Add the query you wrote in step 2.1 to a common table expression called Products. In the outer query, join Products to the Sales.SalesOrderDetail table. Return a list each ProductID with SUM(LineTotal) aliased as ProductTotal. Also include the product, subcategory, category names. Be sure to create an appropriate GROUP BY Clause.

2.3 Move the outer query from 2.2 to a CTE called ProductSales. In the new outer query, create a list of categories with the total sales for each. Name the expression SUM(ProductTotal) “CategoryTotal”.

2.4 Move the outer query from 2.3 to a CTE called CategorySales. In the new outer query, join the CategorySales CTE to the ProductSales CTE on ProductCategory. Now display a list of each ProductId, product name, ProductTotal, Category and CategoryTotal.

2.5 Using the query from 2.4, calculate the percent of each product over the category.

Exercise 3

3.1 Explain the difference between UNION and UNION ALL.

3.2 Write a query that combines the PersonID and the CustomerID from the Sales.Customer table into one column.

3.3 Write a query that returns the list of ProductIDs that have never been part of an order. Do not return any details from the orders, just the list of ProductID and name.