# Lab 7: Using Table Expressions

## Retrieving Product Information

AdventureWorks sells many products that are variants of the same product model. You must write queries that retrieve information about these products.

### Instructions

Retrieve the product ID, product name, product model name, and product model summary for each product from the `SalesLT.Product` table and the `SalesLT.vProductModelCatalogDescription` view. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT P.ProductID, P.Name AS ProductName, PM.Name AS ProductModel, PM.Summary

FROM SalesLT.Product AS P

JOIN SalesLT.vProductModelCatalogDescription AS PM

ON P.ProductModelID = PM.ProductModelID

ORDER BY ProductID;
```

## Retrieving Product Information (2)

You are only interested in products which have a listed color in the database.

**Note: to assist error checking, please just modify the existing code in the editor.**

### Instructions

Create a table variable and populate it with a list of distinct colors from the `SalesLT.Product` table. Then use the table variable to filter a query that returns the product ID, name, and color from the `SalesLT.Product` table so that only products with a color listed in the table variable are returned. You'll need to use `NVARCHAR` in your solution and make sure to use the aliases provided.

```
DECLARE @Colors AS TABLE (Color nvarchar(15));


INSERT INTO @Colors

SELECT DISTINCT Color FROM SalesLT.Product;


SELECT ProductID, Name, Color

FROM SalesLT.Product
```

```
WHERE Color IN (SELECT Color FROM @Colors);
```

## Retrieving Product Information (3)

The `AdventureWorksLT` database includes a table-valued function named `dbo.ufnGetAllCategories`, which returns a table of product categories (e.g. 'Road Bikes') and parent categories (for example 'Bikes').

### Instructions

Write a query that uses this function to return a list of all products including their parent category and their own category. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT C.ParentProductCategoryName AS ParentCategory,
    C.ProductCategoryName AS Category,
    P.ProductID, P.Name AS ProductName
FROM SalesLT.Product AS P
JOIN dbo.ufnGetAllCategories() AS C
ON P.ProductCategoryID = C.ProductCategoryID
ORDER BY ParentCategory, Category, ProductName;
```

## Retrieving Customer Sales Revenue

Each AdventureWorks customer is a retail company with a named contact. You must create queries that return the total revenue for each customer, including the company and customer contact names.

### Instructions

Retrieve a list of customers in the format `Company (Contact Name)` together with the total revenue for each customer. Use a derived table or a common table expression to retrieve the details for each sales order, and then query the derived table or CTE to aggregate and group the data. Make sure to use the aliases provided, and default column names elsewhere.

```
SELECT CompanyContact, SUM(SalesAmount) AS Revenue
FROM
        (SELECT CONCAT(c.CompanyName, CONCAT(' (' + c.FirstName + ' ', c.LastName + ')')), SOH.TotalDue
         FROM SalesLT.SalesOrderHeader AS SOH
        JOIN SalesLT.Customer AS c
        ON SOH.CustomerID = c.CustomerID) AS CustomerSales(CompanyContact,
SalesAmount)
GROUP BY CompanyContact
ORDER BY CompanyContact;
```