Session 3-2 Exercises v03.

Use Starter Kit App "MVCNorthwindProducts01.zip"

ViewModel Questions

ASP.NET CORE Note  
ViewModels need a "[Key]" unique to the data selected by the query  
For some of the following queries you can find existing fields that will do that but they may not be the first fields.  
Mostly you need to write SQL to add this field, usually adding an existing primary key or based on existing primary keys. JPC is naming fields like this "QueryKey".

For each question, analyse the SQL to create a ViewModel – (and add another "QueryKey" field if needed).  
Then use that ViewModel with the SQL to create an on-screen display   
by adding methods and views to the existing "ProductsController".  
HINT, "scaffolding" (generating) a View is often offered as an option in the popup menu we get   
when we right-click on the first statement of a method.

------------

"Customers and Suppliers by City"

SELECT City, CompanyName, ContactName, 'Customers' AS Relationship

FROM Customers

UNION SELECT City, CompanyName, ContactName, 'Suppliers'

FROM Suppliers

This works with "City" as the [Key] field but JPC thinks we got lucky with the data on this one.  
Rewrite with a QueryKey is lots of fun because CustomerId is a string and SupplierId is an integer.  
Therefore convert SupplierId to a string! And not just any string. Best to look up the original database to discover that CustomerId is a nchar(5)

SELECT CustomerId as QueryKey, City, CompanyName, ContactName, 'Customers' AS Relationship

FROM Customers

UNION   
SELECT CONVERT(nchar(5), SupplierId) AS QueryKey, City, CompanyName, ContactName, 'Suppliers'   
 FROM Suppliers

--------=--

"OrderDetails Extended"

SELECT OrderDetails.OrderID, OrderDetails.ProductID, Products.ProductName,

OrderDetails.UnitPrice, OrderDetails.Quantity, OrderDetails.Discount,

(CONVERT(money,(OrderDetails.UnitPrice\*Quantity\*(1-Discount)/100))\*100) AS ExtendedPrice

FROM Products INNER JOIN OrderDetails ON Products.ProductID = OrderDetails.ProductID

This is challenging because "OrderDetails" has a composite Primary Key which we need to duplicate.

SELECT CONVERT(nchar(5), OrderDetails.OrderID) + CONVERT(nchar(5), OrderDetails.ProductID)  
 AS QueryKey,  
 OrderDetails.OrderID, OrderDetails.ProductID, Products.ProductName,

OrderDetails.UnitPrice, OrderDetails.Quantity, OrderDetails.Discount,

(CONVERT(money,(OrderDetails.UnitPrice\*Quantity\*(1-Discount)/100))\*100) AS ExtendedPrice

FROM Products INNER JOIN OrderDetails ON Products.ProductID = OrderDetails.ProductID

NOTE – when we do this kind of rewrite, we can test it quickly in Ms SQL Server Management Studio

-----------------

"Product Sales for 1997"

SELECT Categories.CategoryName, Products.ProductName,

Sum(CONVERT(money,(OrderDetails.UnitPrice\*Quantity\*(1-Discount)/100))\*100) AS ProductSales

FROM (Categories INNER JOIN Products ON Categories.CategoryID = Products.CategoryID)

INNER JOIN (Orders INNER JOIN OrderDetails ON Orders.OrderID = OrderDetails.OrderID)

ON Products.ProductID = OrderDetails.ProductID

WHERE (((Orders.ShippedDate) BETWEEN '1997-01-01' AND '1997-12-31'))

GROUP BY Categories.CategoryName, Products.ProductName

I thought at first look that this was all about OrderDetails and I tried the composite QueryKey from above.  
That crashed in Management Studio. Looking at this more carefully, I see this is really about a report on each product. The GROUP BY OrderDetails means that we get OrderDetails compressed to one line of information for each Product.

Therefore the [Key] goes on the second field "ProductName"

We check this by copying and pasting this SQL into Microsoft SQL Server Management Studio.  
Sure enough, we can see each ProductName appearing only once

-----------------

"Ten Most Expensive Products"

This is relatively easy. It is table "Products" with a little more text information from "Categories"  
Therefore the first field ProductId is a Primary Key and we can label it [Key] with no need to rewrite the SQL

SELECT TOP 10 Products.ProductId, Products.ProductName, Categories.CategoryName, Products.UnitPrice, Products.QuantityPerUnit

FROM Products INNER JOIN Categories ON Products.CategoryId = Categories.CategoryId

ORDER BY Products.UnitPrice DESC

-------------------

**Search Example**  
Provide a Search on the ProductName field of table Products.  
JPC will demonstrate this in class before you do the activity.

Example code:

Achieved by modifying method "Index" in "ProductsControllers":

public ActionResult Index(string searchString)

{

string sql = "SELECT \* FROM Products WHERE ProductName LIKE @p0";

searchString = "%" + searchString + "%";

List<Product> products = db.Products.SqlQuery(sql, searchString).ToList();

return View(products);

}

Note the need in the above code to use LIKE rather than the equals sign.  
In a separate line of code we add the wildcard "%" before and after searchString   
to specify that we are looking for ProductNames which contain searchString.

Test with this test URL

GET Products/Index?searchString=gum

Note that the "routing" alternative URL style is not working here.

(NOT working) GET Products/Index/gum

-----------------------------------------------------------------------------------------------

Search Questions

Provide a Search on the QuantityPerUnit of table Products  
You can adapt the above ProductsController example for this.

Provide a Search on both FirstName and LastName fields of table Employees  
You will need a new controller for this.

Provide a Search on the CompanyName field of table Customers  
You will need a new controller for this.