Josh Wolfe

AdventureWorks Product Sales Analysis

Objectives

1. Provide a detailed list of Internet sales with the following columns for the financial analyst team to review (Category, Model, CustomerKey, Region, IncomeGroup, CalendarYear, FiscalYear, Month, OrderNumber, Quantity, and Amount). Income group should categorize the people based on "Low" being less than 40,000, "High" being greater than 60,000, and the rest will be "Moderate".

```
SELECT
pc.EnglishProductCategoryName, p.EnglishProductName, fis.CustomerKey,
st.SalesTerritoryRegion,
	CASE
		WHEN c.YearlyIncome < 40000.00 THEN 'Low'
		WHEN c.YearlyIncome > 60000.00 THEN 'High'
		ELSE 'Medium'
	END AS IncomeGroup,
d.CalendarYear, d.FiscalYear, d.EnglishMonthName,
fis.SalesOrderNumber, fis.OrderQuantity, fis.SalesAmount

FROM FactInternetSales AS fis
INNER JOIN product AS p
ON fis.ProductKey = p.ProductKey
LEFT JOIN ProductSubcategory AS psc
ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
LEFT JOIN ProductCategory AS pc
ON psc.ProductCategoryKey = pc.ProductCategoryKey
LEFT JOIN SalesTerritory AS st
ON fis.SalesTerritoryKey = st.SalesTerritoryKey
LEFT JOIN Customer AS c
ON fis.CustomerKey = c.CustomerKey
LEFT JOIN Date AS d
ON fis.OrderDateKey = d.DateKey
ORDER BY fis.SalesOrderNumber
```

   a. This query joins (both internally and externally) seven tables to return the requested fields; Category, Model, CustomerKey, Region, IncomeGroup, CalendarYear, FiscalYear, Month, OrderNumber, Quantity, and Amount.

      The first table is FactInternetSales and is inner joined with the Products table. The use of inner join in this case ensures only orders that contain products, and only products that have been ordered, are returned instead of either empty orders or products that have yet to be ordered. These tables are joined on the ProductKey. The FactInternetSales table returns the CustomerKey, OrderQuantity, and SalesAmount fields while the Product table returns the EnglishProductName (Model).

      The ProductSubCategory and ProductCategory are Left Joined in respective order. The left join returns only categories that have products sold. The ProductSubCategory table has a primary key “ProductSubcategoryKey” that is a foreign key in the Products table.

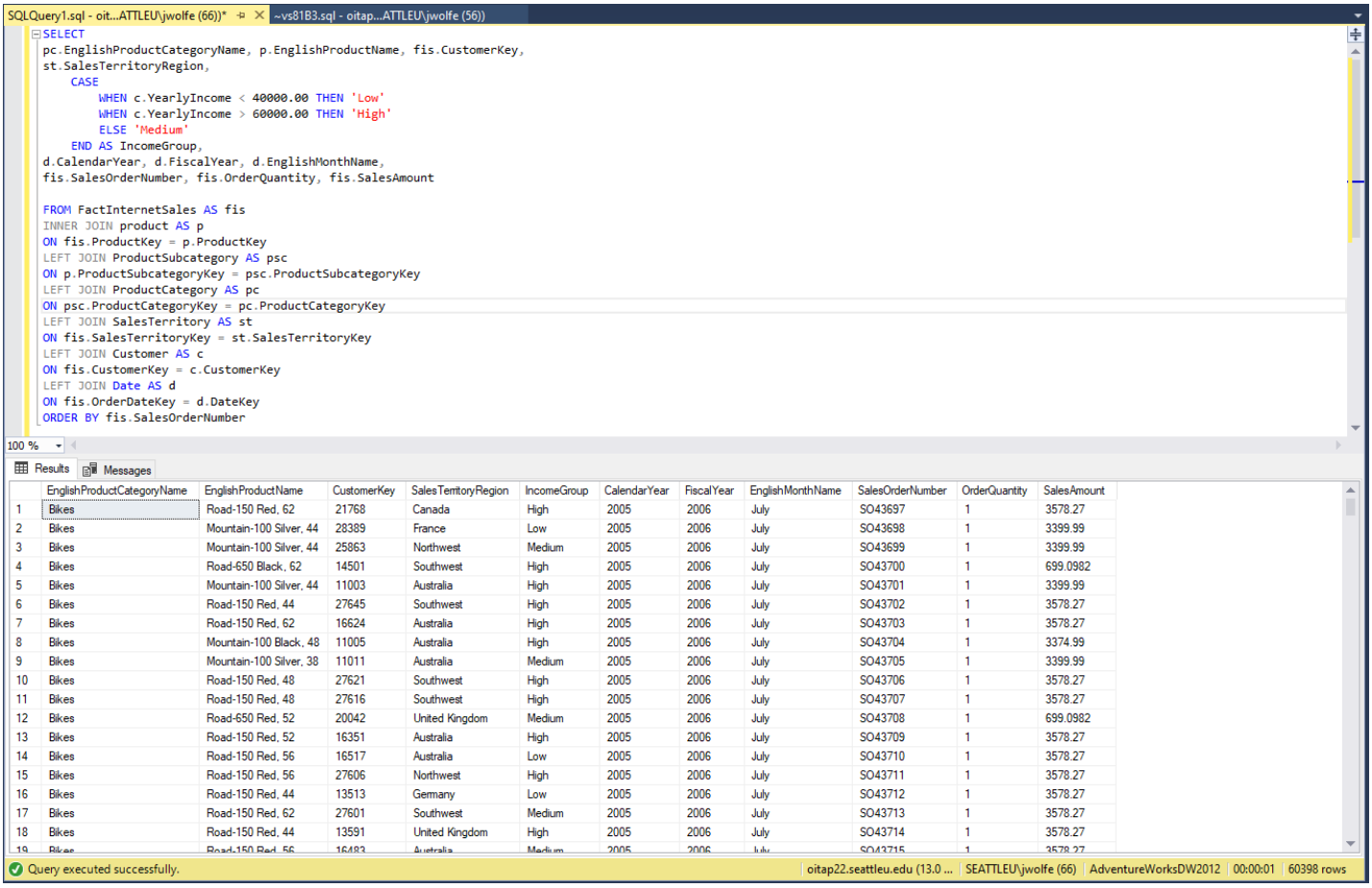
The ProductCategory table has a primary key “ProductCategoryKey” found in the ProductSubCategory table as a foreign key. This relationship between these three tables is how the query links the Product table to the ProductCategory table. The EnglishProductCategoryName (Model) field is returned by the ProductCategory table.

The SalesTerritory table is merged through a left join to ensure only territories related to the sales are returned. The primary key SalesTerritoryKey in the SalesTerritory table is a foreign key in the FactInternetSales table and is used for the join. The SalesTerritory table returns the SalesTerritoryRegion (Region) field.

The Customer table is left joined with the FactInternetSales table through the primary/foreign key CustomerKey. This left join ensures only customers related to these specific internet sales are included in the analysis. The Customer table returns the IncomeGroup information.

The final table merged is the Date table. This table is left joined using the primary key DateKey in the Date table and the foreign key OrderDateKey in the FactInternetSales table. The Date table returns the CalendarYear, FiscalYear, and EnglishMonthName (Month) fields.

The IncomeGroup provided by the Customer table is generated by using CASE in the SELECT clause. The customers are broken up into three income groups based on the given parameters of annual income greater than 60,000 as High, lower than 40,000 as Low, and everything in-between as Middle. The CASE is terminated as IncomeGroup. In an effort to organize the results they are ordered by the SalesOrderNumber starting with the smallest order number at the top.



```
SELECT
pc.EnglishProductCategoryName, p.EnglishProductName, fis.CustomerKey,
st.SalesTerritoryRegion,
    CASE
        WHEN c.YearlyIncome < 40000.00 THEN 'Low'
        WHEN c.YearlyIncome > 60000.00 THEN 'High'
        ELSE 'Medium'
    END AS IncomeGroup,
d.CalendarYear, d.FiscalYear, d.EnglishMonthName,
fis.SalesOrderNumber, fis.OrderQuantity, fis.SalesAmount

FROM FactInternetSales AS fis
INNER JOIN product AS p
ON fis.ProductKey = p.ProductKey
LEFT JOIN ProductSubcategory AS psc
ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
LEFT JOIN ProductCategory AS pc
ON psc.ProductCategoryKey = pc.ProductCategoryKey
LEFT JOIN SalesTerritory AS st
ON fis.SalesTerritoryKey = st.SalesTerritoryKey
LEFT JOIN Customer AS c
ON fis.CustomerKey = c.CustomerKey
LEFT JOIN Date AS d
ON fis.OrderDateKey = d.DateKey
ORDER BY fis.SalesOrderNumber
```

	EnglishProductCategoryName	EnglishProductName	CustomerKey	SalesTerritoryRegion	IncomeGroup	CalendarYear	FiscalYear	EnglishMonthName	SalesOrderNumber	OrderQuantity	SalesAmount
1	Bikes	Road-150 Red, 62	21768	Canada	High	2005	2006	July	SO43697	1	3578.27
2	Bikes	Mountain-100 Silver, 44	28389	France	Low	2005	2006	July	SO43698	1	3399.99
3	Bikes	Mountain-100 Silver, 44	25863	Northwest	Medium	2005	2006	July	SO43699	1	3399.99
4	Bikes	Road-650 Black, 62	14501	Southwest	High	2005	2006	July	SO43700	1	699.0982
5	Bikes	Mountain-100 Silver, 44	11003	Australia	High	2005	2006	July	SO43701	1	3399.99
6	Bikes	Road-150 Red, 44	27645	Southwest	High	2005	2006	July	SO43702	1	3578.27
7	Bikes	Road-150 Red, 62	16624	Australia	High	2005	2006	July	SO43703	1	3578.27
8	Bikes	Mountain-100 Black, 48	11005	Australia	High	2005	2006	July	SO43704	1	3374.99
9	Bikes	Mountain-100 Silver, 38	11011	Australia	Medium	2005	2006	July	SO43705	1	3399.99
10	Bikes	Road-150 Red, 48	27621	Southwest	High	2005	2006	July	SO43706	1	3578.27
11	Bikes	Road-150 Red, 48	27616	Southwest	High	2005	2006	July	SO43707	1	3578.27
12	Bikes	Road-650 Red, 52	20042	United Kingdom	Medium	2005	2006	July	SO43708	1	699.0982
13	Bikes	Road-150 Red, 52	16351	Australia	High	2005	2006	July	SO43709	1	3578.27
14	Bikes	Road-150 Red, 56	16517	Australia	Low	2005	2006	July	SO43710	1	3578.27
15	Bikes	Road-150 Red, 56	27606	Northwest	High	2005	2006	July	SO43711	1	3578.27
16	Bikes	Road-150 Red, 44	13513	Germany	Low	2005	2006	July	SO43712	1	3578.27
17	Bikes	Road-150 Red, 62	27601	Southwest	Medium	2005	2006	July	SO43713	1	3578.27
18	Bikes	Road-150 Red, 44	13591	United Kingdom	High	2005	2006	July	SO43714	1	3578.27

2. Provide a similar analysis for Reseller sales with the following columns (Category, Model, CalendarYear, FiscalYear, Month, OrderNumber, Quantity, Amount).

```
SELECT
pc.EnglishProductCategoryName, p.EnglishProductName,
d.CalendarYear, d.FiscalYear, d.EnglishMonthName,
rss.SalesOrderNumber, rss.OrderQuantity, rss.SalesAmount

FROM FactResellerSales AS rss
INNER JOIN product AS p
ON rss.ProductKey = p.ProductKey
LEFT JOIN ProductSubcategory AS psc
ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
LEFT JOIN ProductCategory AS pc
ON psc.ProductCategoryKey = pc.ProductCategoryKey
LEFT JOIN Date AS d
ON rss.OrderDateKey = d.DateKey
ORDER BY pc.EnglishProductCategoryName, p.EnglishProductName
```

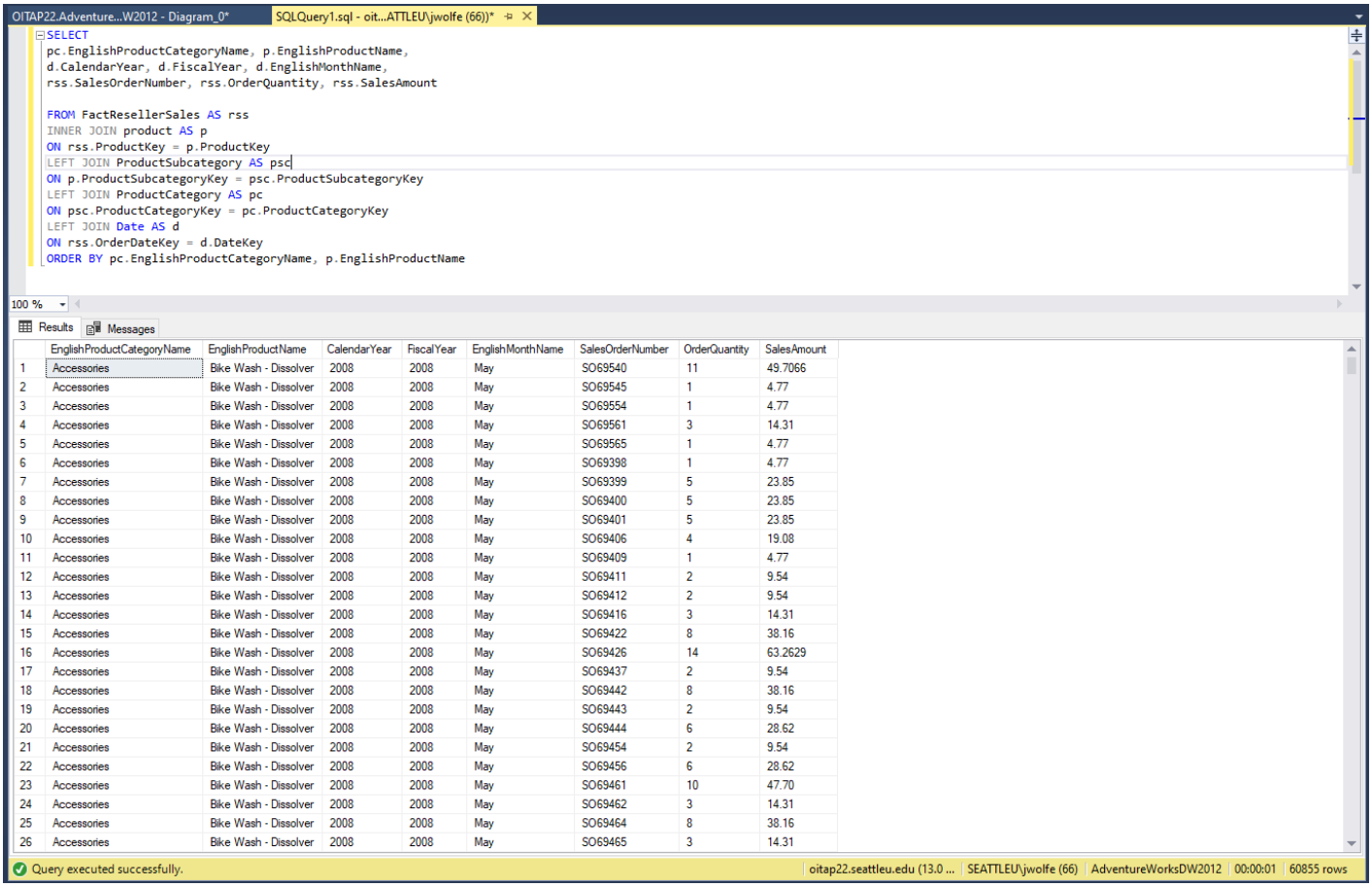
   a. This query is similar to the previous except it is based on the FactResellerSales table instead of the FactInternetSales table. There are 5 tables joined using both internal and external methods. The query returns the Category (EnglishProductCategoryName), Model (EnglishProductName), CalendarYear, FiscalYear, Month (EnglishMonthName), OrderNumber (SalesOrderNumber), Quantity (OrderQuantity), and Amount (Sales

Amount) fields.

The first table is FactResellerSales. This table returns the SalesOrderNumber (OrderNumber), OrderQuantity (Quantity), and SalesAmount (Amount) fields for all sales done through resellers. This table is inner joined with the Product table to return the EnglishProductName (Model) field for each product ordered. The inner join ensures that only orders with products (avoiding empty orders) and products purchased (avoiding non-purchased products) are returned. These two tables are joined using the primary key ProductKey found in the Product table with the foreign key ProductKey found in the FactResellerSales table.

The ProductSubcategory table is used to merge the Product table and the ProductCategory table. The ProductSubcategory table is left joined to the Product table using the ProductSubcategoryKey and the ProductCategory table is left joined with the ProductSubcategory table on the ProductCategoryKey. The left joining in both cases ensure that neither a subcategory or category that have no relation to the products sold are returned. The ProductCategoryKey returns the EnglishProductCategoryName (Category) field.

The last Date table is the final table to be merged and is done so using a left join with the FactResellerSales table on the foreign OrderDateKey to primary DateKey. The Date table returns the CalenderYear, FiscalYear, and EnglishMonthName (Month) fields. The query is then sorted in alphabetical order by category name followed by model name.



	EnglishProductCategoryName	EnglishProductName	CalendarYear	FiscalYear	EnglishMonthName	SalesOrderNumber	OrderQuantity	SalesAmount
1	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69540	11	49.7066
2	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69545	1	4.77
3	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69554	1	4.77
4	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69561	3	14.31
5	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69565	1	4.77
6	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69398	1	4.77
7	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69399	5	23.85
8	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69400	5	23.85
9	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69401	5	23.85
10	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69406	4	19.08
11	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69409	1	4.77
12	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69411	2	9.54
13	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69412	2	9.54
14	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69416	3	14.31
15	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69422	8	38.16
16	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69426	14	63.2629
17	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69437	2	9.54
18	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69442	8	38.16
19	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69443	2	9.54
20	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69444	6	28.62
21	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69454	2	9.54
22	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69456	6	28.62
23	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69461	10	47.70
24	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69462	3	14.31
25	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69464	8	38.16
26	Accessories	Bike Wash - Dissolver	2008	2008	May	SO69465	3	14.31

3. Show the total sales (overall) by year rolled up by the Territory group and country. A special request is that the United Kingdom is no longer part of Europe and management wants to see their totals as a separate Territory group. You cannot modify the data, so you will need to address this request in your query.

```
SELECT sq.Year,
	CASE
		WHEN GROUPING(sq.SalesTerritoryGroup) = 1
		THEN '-All Territories-'
		ELSE sq.SalesTerritoryGroup
	END AS TerritoryGroup,
	CASE
		WHEN GROUPING(sq.SalesTerritoryCountry) = 1
		THEN '-All Countries-'
		ELSE sq.SalesTerritoryCountry
	END AS Country,
ROUND(SUM(sq.SalesAmount), 2) AS SalesTotals

FROM (
	SELECT YEAR(fis.OrderDate) AS Year, st.SalesTerritoryGroup,
st.SalesTerritoryCountry, fis.SalesAmount
	FROM FactInternetSales AS fis
	INNER JOIN SalesTerritory AS st
	ON fis.SalesTerritoryKey = st.SalesTerritoryKey

	UNION
```

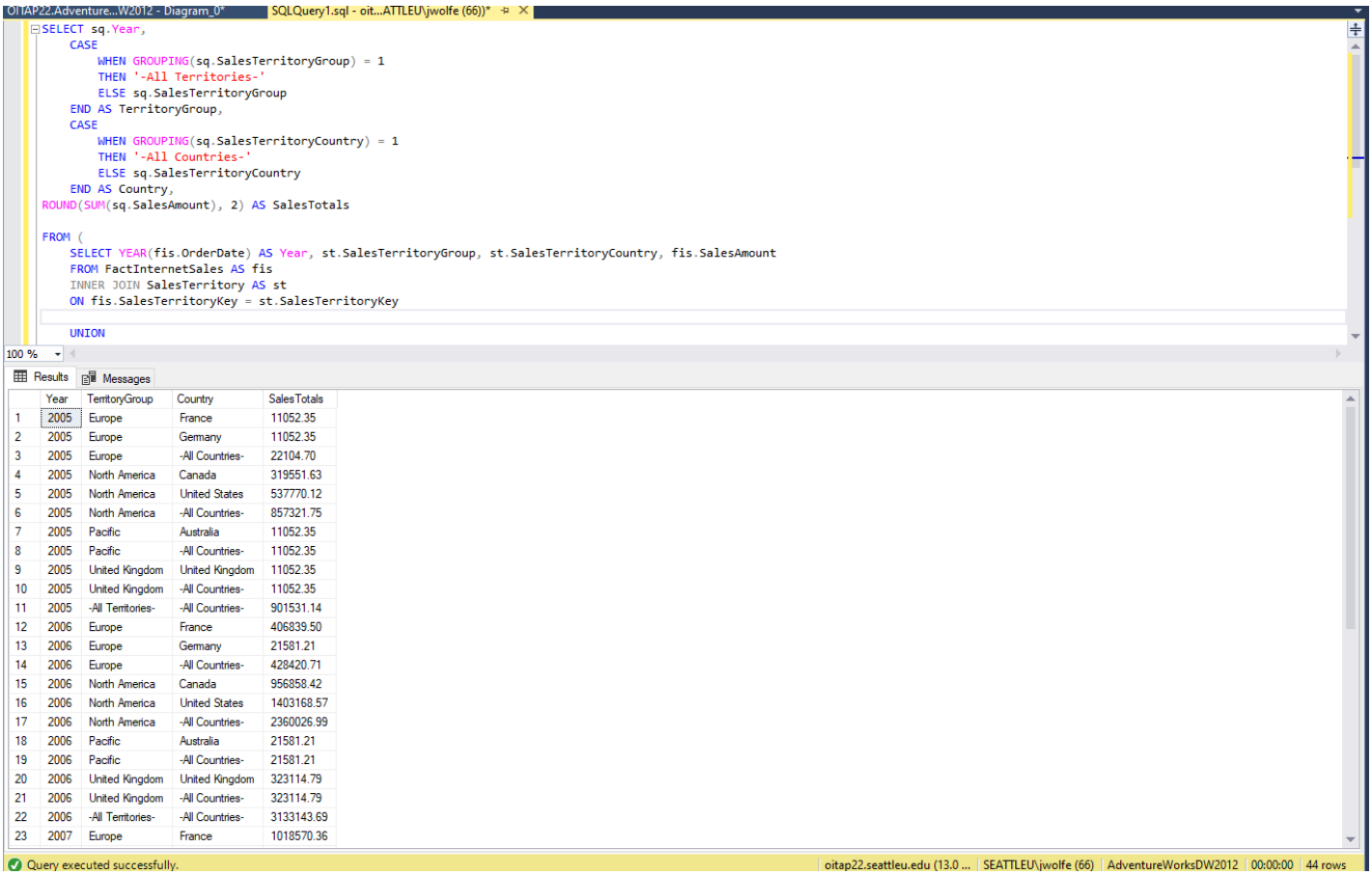
```
        SELECT YEAR(rss.OrderDate) AS Year, st.SalesTerritoryGroup,
st.SalesTerritoryCountry, rss.SalesAmount
        FROM FactResellerSales AS rss
        INNER JOIN SalesTerritory AS st
        ON rss.SalesTerritoryKey = st.SalesTerritoryKey) AS sq

GROUP BY
        sq.Year,
        ROLLUP(sq.SalesTerritoryGroup, sq.SalesTerritoryCountry)
```

a. This query returns the total sales (SalesTotals) for each country, each region (TerritoryGroup), and overall. The query is organized in ascending order base on year. The returned fields are Year, TerritoryGroup, Country, and SalesTotals. The query is characterized by three main components, a CASE statement in the SELECT clause, a FROM subquery containing a UNION to join all sales methods, and a GROUP BY/ROLLUP clause to organize the data by year, region, and country respectively.

   The main component is the union in the middle of the query. Sales are held in two separate tables, FactInternetSales and FactResellerSales. These tables are stacked while retaining only the year of the order (using a YEAR() function and the OrderDate field), the sales amount, and the territory information (gathered from a join with the SalesTerritory tables using the SalesTerritoryKey). This union is contained in a FROM clause subquery so that a math function (SUM) can be used in the top-level SELECT clause.

   The SELECT clause returns the subquery Year field, a clause statement, and a math function to total the sales amounts. The SUM() function is used to total all sale amounts grouped on the country, territory, and overall. The results are then rounded used the ROUND() function to two decimal places. The CASE statement is used to remove the NULL values returned by the GROUP BY/ROLLUP in the rolled up fields. An example is when the table is displaying the overall sales results for the Europe territory, instead of the Country field displaying NULL, the field will display –All Countries-. It works similarly for all territories when retuning the yearly totals.

   The GROUP BY/ROLLUP clause is what brings this entire table together. The data is grouped by the Year field, then rolled up by the SalesTerritoryGroup and the SalesTerritoryCountry. The table is in acedning order with the oldest year at the top. Then the table is broken down by each territory, and then further broken down by each country. The sales totals are given for each country, each territory, and each year including a grand total for all of the years.



```
SELECT sq.Year,
    CASE
        WHEN GROUPING(sq.SalesTerritoryGroup) = 1
        THEN '-All Territories-'
        ELSE sq.SalesTerritoryGroup
    END AS TerritoryGroup,
    CASE
        WHEN GROUPING(sq.SalesTerritoryCountry) = 1
        THEN '-All Countries-'
        ELSE sq.SalesTerritoryCountry
    END AS Country,
ROUND(SUM(sq.SalesAmount), 2) AS SalesTotals

FROM (
    SELECT YEAR(fis.OrderDate) AS Year, st.SalesTerritoryGroup, st.SalesTerritoryCountry, fis.SalesAmount
    FROM FactInternetSales AS fis
    INNER JOIN SalesTerritory AS st
    ON fis.SalesTerritoryKey = st.SalesTerritoryKey

    UNION
```

Results | Messages

	Year	TerritoryGroup	Country	SalesTotals
1	2005	Europe	France	11052.35
2	2005	Europe	Germany	11052.35
3	2005	Europe	-All Countries-	22104.70
4	2005	North America	Canada	319551.63
5	2005	North America	United States	537770.12
6	2005	North America	-All Countries-	857321.75
7	2005	Pacific	Australia	11052.35
8	2005	Pacific	-All Countries-	11052.35
9	2005	United Kingdom	United Kingdom	11052.35
10	2005	United Kingdom	-All Countries-	11052.35
11	2005	-All Territories-	-All Countries-	901531.14
12	2006	Europe	France	406839.50
13	2006	Europe	Germany	21581.21
14	2006	Europe	-All Countries-	428420.71
15	2006	North America	Canada	956858.42
16	2006	North America	United States	1403168.57
17	2006	North America	-All Countries-	2360026.99
18	2006	Pacific	Australia	21581.21
19	2006	Pacific	-All Countries-	21581.21
20	2006	United Kingdom	United Kingdom	323114.79
21	2006	United Kingdom	-All Countries-	323114.79
22	2006	-All Territories-	-All Countries-	3133143.69
23	2007	Europe	France	1018570.36

Query executed successfully. | oitap22.seattleu.edu (13.0 ... | SEATTLEU\jwolfe (66) | AdventureWorksDW2012 | 00:00:00 | 44 rows

4. Provide an analysis of sales performance by Promotion. It would be interesting to see how different types of promotions drive sales (quantity and revenue), especially by product category or region. The comparison between Internet and Reseller sales is probably interesting too. Don't attempt to do everything, but show some good analysis related to Promotion.

```
SELECT sq.EnglishProductCategoryName,
p.EnglishPromotionName,

        CASE
                WHEN GROUPING(st.SalesTerritoryGroup) = 1
                THEN '-All Territories-'
                ELSE st.SalesTerritoryGroup
        END AS TerritoryGroup,
        CASE
                WHEN GROUPING(st.SalesTerritoryCountry) = 1
                THEN '-All Countries-'
                ELSE st.SalesTerritoryCountry
        END AS Country,

ROUND(SUM(sq.OrderQuantity), 2) AS OrderQuantity,
ROUND(SUM(sq.SalesAmount), 2) AS Revenue

FROM Promotion AS p
INNER JOIN (
        SELECT fis.ProductKey, fis.PromotionKey, fis.SalesAmount, fis.OrderQuantity,
```

```
	pr.EnglishProductSubcategoryName, pr.EnglishProductCategoryName,
fis.SalesTerritoryKey
		FROM FactInternetSales AS fis

		INNER JOIN (
			SELECT p.ProductKey, psc.EnglishProductSubcategoryName,
			pc.EnglishProductCategoryName
			FROM product AS p
			INNER JOIN ProductSubcategory AS psc
			ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
			INNER JOIN ProductCategory AS pc
			ON psc.ProductCategoryKey = pc.ProductCategoryKey) AS pr

		ON pr.ProductKey = fis.ProductKey

		UNION

		SELECT rss.ProductKey, rss.PromotionKey, rss.SalesAmount,
rss.OrderQuantity,
		pr.EnglishProductSubcategoryName, pr.EnglishProductSubcategoryName,
rss.SalesTerritoryKey
		FROM FactResellerSales AS rss

		INNER JOIN (
			SELECT p.ProductKey, psc.EnglishProductSubcategoryName,
			pc.EnglishProductCategoryName
			FROM product AS p
			INNER JOIN ProductSubcategory AS psc
			ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
			INNER JOIN ProductCategory AS pc
			ON psc.ProductCategoryKey = pc.ProductCategoryKey) AS pr

		ON rss.ProductKey = pr.ProductKey) AS sq

ON p.PromotionKey = sq.PromotionKey

INNER JOIN SalesTerritory AS st
ON sq.SalesTerritoryKey = st.SalesTerritoryKey

GROUP BY sq.EnglishProductCategoryName, p.EnglishPromotionName,
	ROLLUP(st.SalesTerritoryGroup,
	st.SalesTerritoryCountry)

ORDER BY sq.EnglishProductCategoryName
```

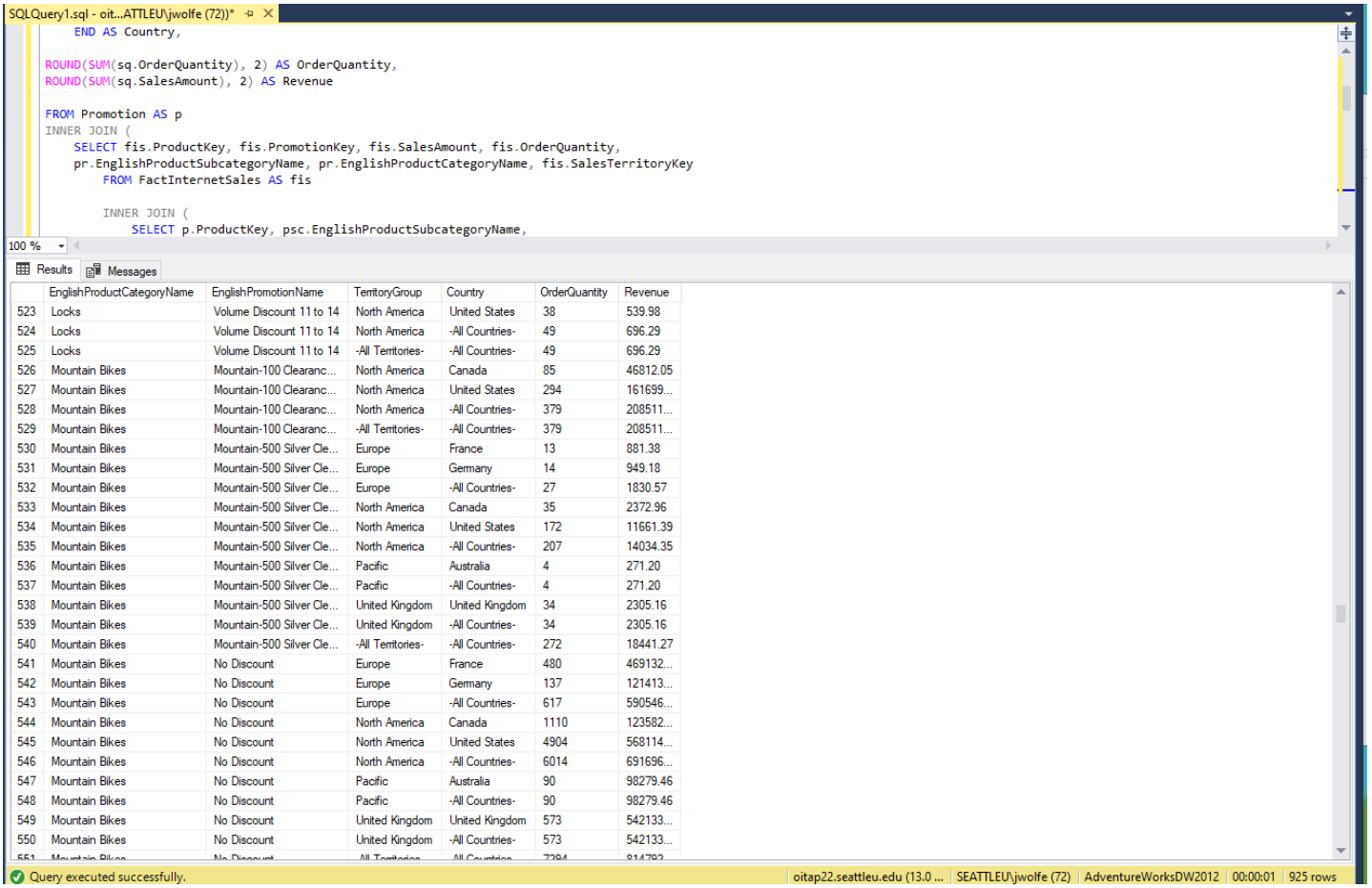
a. The concept and approach of this query is similar to the last. The general idea is that revenue and order quantity is totaled for each category, promotion type, region, and country. The main idea is to examine how promotion sales look in comparison to non-promotion sales, and it further broken down by region and country. The query returns the Category (EnglishProductCategoryName), Promotion (EnglishPromotionName), TerritoryGroup, Country, OrderQuantity, and Revenue fields.

   The primary component of this query is the joining of a subquery to the Promotion

table. This subquery is a union of two more subqueries that combine tables for internet and reseller sales. Both of these tables are joined in separate (sub)subqueries to pull the product's category information from the category tables before the union. The original subquery collects the ProductKey, PromotionKey, SalesAmount, OrderQuantity, SalesTerritoyKey, and product category fields so that they can be called in the top-level SELECT clause.

In addition to the category and promotion fields, the SELECT clause contains a CASE statement that aids in clarifying the table where the total revenue and sales for promotion, region, and country are returned. The SELECT clause also uses two math functions to sum the OrderQuantity and SalesAmount. Both of these functions are rounded to two decimal places.

The GROUP BY/ROLLUP is what categorizes the returned table. The results are grouped by category followed by promotion name. Grouping by category first allows for easy comparison of how promotion are related to sales within any given category. The TerritoyGroup and Country are rolled up so that the promotions can be measured against each other depending on region and country, including the overall totals. The table is also sorted by category name in ascending order to aid in organization.



	EnglishProductCategoryName	EnglishPromotionName	TerritoryGroup	Country	OrderQuantity	Revenue
523	Locks	Volume Discount 11 to 14	North America	United States	38	539.98
524	Locks	Volume Discount 11 to 14	North America	-All Countries-	49	696.29
525	Locks	Volume Discount 11 to 14	-All Territories-	-All Countries-	49	696.29
526	Mountain Bikes	Mountain-100 Clearanc...	North America	Canada	85	46812.05
527	Mountain Bikes	Mountain-100 Clearanc...	North America	United States	294	161699...
528	Mountain Bikes	Mountain-100 Clearanc...	North America	-All Countries-	379	208511...
529	Mountain Bikes	Mountain-100 Clearanc...	-All Territories-	-All Countries-	379	208511...
530	Mountain Bikes	Mountain-500 Silver Cle...	Europe	France	13	881.38
531	Mountain Bikes	Mountain-500 Silver Cle...	Europe	Germany	14	949.18
532	Mountain Bikes	Mountain-500 Silver Cle...	Europe	-All Countries-	27	1830.57
533	Mountain Bikes	Mountain-500 Silver Cle...	North America	Canada	35	2372.96
534	Mountain Bikes	Mountain-500 Silver Cle...	North America	United States	172	11661.39
535	Mountain Bikes	Mountain-500 Silver Cle...	North America	-All Countries-	207	14034.35
536	Mountain Bikes	Mountain-500 Silver Cle...	Pacific	Australia	4	271.20
537	Mountain Bikes	Mountain-500 Silver Cle...	Pacific	-All Countries-	4	271.20
538	Mountain Bikes	Mountain-500 Silver Cle...	United Kingdom	United Kingdom	34	2305.16
539	Mountain Bikes	Mountain-500 Silver Cle...	United Kingdom	-All Countries-	34	2305.16
540	Mountain Bikes	Mountain-500 Silver Cle...	-All Territories-	-All Countries-	272	18441.27
541	Mountain Bikes	No Discount	Europe	France	480	469132...
542	Mountain Bikes	No Discount	Europe	Germany	137	121413...
543	Mountain Bikes	No Discount	Europe	-All Countries-	617	590546...
544	Mountain Bikes	No Discount	North America	Canada	1110	123582...
545	Mountain Bikes	No Discount	North America	United States	4904	568114...
546	Mountain Bikes	No Discount	North America	-All Countries-	6014	691696...
547	Mountain Bikes	No Discount	Pacific	Australia	90	98279.46
548	Mountain Bikes	No Discount	Pacific	-All Countries-	90	98279.46
549	Mountain Bikes	No Discount	United Kingdom	United Kingdom	573	542133...
550	Mountain Bikes	No Discount	United Kingdom	-All Countries-	573	542133...

5. Our customers are always a big discussion topic with management and the sales team. The Customer table has a wealth of data categories that could be joined with Internet sales and all the extra data that brings along. This request will likely separate the high-performing analysts from the rest.

```
SELECT sq.CustomerKey,
sq.BikeOrders, sq.ClothingOrders, sq.AccessoryOrders, sq.TotalOrders,
SUM(cs.CoupleSize + c.TotalChildren) AS FamilySize, sq.TotalSales,
FORMAT(SUM((sq.BikeOrders*1.0 / sq.TotalOrders*1.0)*100), '#0.00') AS PercBikeOrders,
FORMAT(SUM((sq.ClothingOrders*1.0 / sq.TotalOrders*1.0)*100), '#0.00') AS
PercClothOrders,
FORMAT(SUM((sq.AccessoryOrders*1.0 / sq.TotalOrders*1.0)*100), '#0.00') AS PercAccOrders

FROM (
	SELECT DISTINCT c.CustomerKey, TotalOrders.TotalSales,
	ISNULL(bikes.BikeOrders, 0) AS BikeOrders,
	ISNULL(Accessories.AccessoryOrders, 0) AS AccessoryOrders,
	ISNULL(Clothing.ClothingOrders, 0) AS ClothingOrders,
	ISNULL(TotalOrders.TotalOrders, 0) AS TotalOrders
	FROM FactInternetSales AS fis

	LEFT JOIN Customer AS c
	ON fis.CustomerKey = c.CustomerKey

	LEFT JOIN (
		SELECT c.CustomerKey, COUNT(fis.SalesOrderNumber) AS BikeOrders
		FROM FactInternetSales AS fis
		INNER JOIN product AS p
		ON p.ProductKey = fis.ProductKey
		INNER JOIN ProductSubcategory AS psc
		ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
		INNER JOIN ProductCategory AS pc
		ON psc.ProductCategoryKey = pc.ProductCategoryKey
		INNER JOIN Customer AS c
		ON fis.CustomerKey = c.CustomerKey
		WHERE pc.ProductCategoryKey = 1
		GROUP BY c.CustomerKey) AS Bikes
	ON fis.CustomerKey = Bikes.CustomerKey

	LEFT JOIN (
		SELECT c.CustomerKey, COUNT(fis.SalesOrderNumber) AS ClothingOrders
		FROM FactInternetSales AS fis
		INNER JOIN product AS p
		ON p.ProductKey = fis.ProductKey
		INNER JOIN ProductSubcategory AS psc
		ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
		INNER JOIN ProductCategory AS pc
		ON psc.ProductCategoryKey = pc.ProductCategoryKey
		INNER JOIN Customer AS c
		ON c.CustomerKey = fis.CustomerKey
		WHERE pc.ProductCategoryKey = 3
		GROUP BY c.CustomerKey) AS Clothing
	ON fis.CustomerKey = Clothing.CustomerKey

	LEFT JOIN (
		SELECT c.CustomerKey, COUNT(fis.SalesOrderNumber) AS AccessoryOrders
		FROM FactInternetSales AS fis
```

```
                INNER JOIN product AS p
                ON p.ProductKey = fis.ProductKey
                INNER JOIN ProductSubcategory AS psc
                ON p.ProductSubcategoryKey = psc.ProductSubcategoryKey
                INNER JOIN ProductCategory AS pc
                ON psc.ProductCategoryKey = pc.ProductCategoryKey
                INNER JOIN Customer AS c
                ON c.CustomerKey = fis.CustomerKey
                WHERE pc.ProductCategoryKey = 4
                GROUP BY c.CustomerKey ) AS Accessories
        ON fis.CustomerKey = Accessories.CustomerKey

        LEFT JOIN (
                SELECT c.CustomerKey,
                SUM(fis.SalesAmount) AS TotalSales,
                COUNT(fis.SalesOrderNumber) AS TotalOrders
                FROM FactInternetSales AS fis
                INNER JOIN Customer AS c
                ON fis.CustomerKey = c.CustomerKey
                GROUP BY c.CustomerKey ) AS TotalOrders
        ON fis.CustomerKey = TotalOrders.CustomerKey) AS sq

INNER JOIN Customer AS c
ON sq.CustomerKey = c.CustomerKey

INNER JOIN (
        SELECT c.CustomerKey,
                CASE
                WHEN c.MaritalStatus = 'M' THEN 2
                WHEN c.MaritalStatus = 'S' THEN 1
                ELSE 0
                END AS CoupleSize
        FROM Customer AS c) AS cs
ON c.CustomerKey = cs.CustomerKey

GROUP BY sq.CustomerKey, sq.BikeOrders, sq.ClothingOrders,
sq.AccessoryOrders, sq.TotalOrders, sq.TotalSales

ORDER BY sq.CustomerKey
```

a. This query takes customer information from the Customer table and combines it with the FactInternetSales table to examine what customers order from products from which categories, along with their order count and total sales. The idea is to target marketing to customers based on sales history. For example, if a customer is purchasing a lot of products from the Bike category, the company could send a discount/coupon for products in the accessory category for items that complement their prior purchases. This could potentially encourage more sales. The customer family size is also included with a similar thought process. If a customer has a couple bike purchases but a family of six, then discounts can be sent out to encourage purchasing more bikes or accompanying products for other family members.

   The main component of this query is calculating the order counts for the desired category groups (Bikes, Clothing, Accessories). There are three filtering subqueries to calculate the orders under the Bike, Clothing, and Accessories category as well as one to

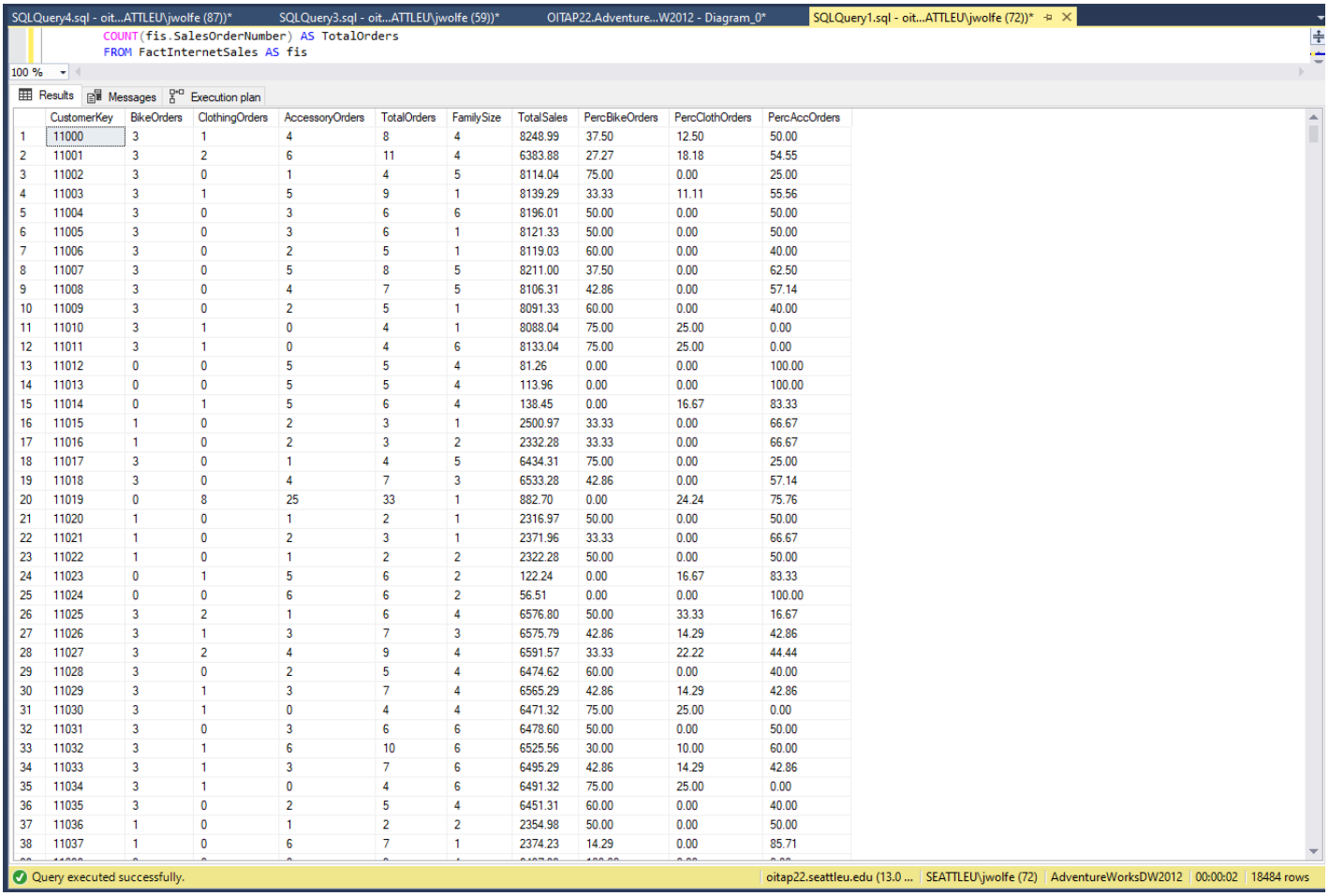
calculate the total amount of orders across all categories. These subqueries are all joined together using the CustomerKey found in the Customer table.

These four subqueries are wrapped up into another subquery in the FROM clause for the top-level SELECT clause. This subquery returns the CustomerKey, TotalSales, BikeOrders, AccessoryOrders, ClothingOrders, and TotalOrders fields. The ISNULL() function is used on the three category calculations to replace the null fields (those that had zero orders) with zeros for later calculations.

The subquery in the top-level FROM clause is inner joined with both the Customer table to return the customer ID and another subquery table that contains a CASE statement. The CASE statement is used to turn the MartialStatus field found in the Customer table into a 1, or 2 (single, married, counting for each person in the couple) to later be combined with the TotalChilden field, creating the FamilySize field.

The top-level SELECT clause returns the CustomerKey, BikeOrders, ClothingOrders, AccessoryOrders, TotalOrders, FamilySize, TotalSales, PercBikeOrders, PercClothOrders, and PercAccOrders fields. There are four math functions used to return the FamilySize and the percentage of orders for the three category fields. FamilySize takes the sum of the CoupleSize generated by the CASE statement and the TotalChildren field from the Customer table. The categories use the SUM() function with the order count divided by the total order count and multiplied by 100 to get the percentage. The Format() function is used to round off the trailing zeros.

The GROUP BY clause returns the non-aggregate expressions in the top-level SELECT clause. The entire table is ordered by the Customerkey in ascending order.



	CustomerKey	BikeOrders	ClothingOrders	AccessoryOrders	TotalOrders	FamilySize	TotalSales	PercBikeOrders	PercClothOrders	PercAccOrders
1	11000	3	1	4	8	4	8248.99	37.50	12.50	50.00
2	11001	3	2	6	11	4	6383.88	27.27	18.18	54.55
3	11002	3	0	1	4	5	8114.04	75.00	0.00	25.00
4	11003	3	1	5	9	1	8139.29	33.33	11.11	55.56
5	11004	3	0	3	6	6	8196.01	50.00	0.00	50.00
6	11005	3	0	3	6	1	8121.33	50.00	0.00	50.00
7	11006	3	0	2	5	1	8119.03	60.00	0.00	40.00
8	11007	3	0	5	8	5	8211.00	37.50	0.00	62.50
9	11008	3	0	4	7	5	8106.31	42.86	0.00	57.14
10	11009	3	0	2	5	1	8091.33	60.00	0.00	40.00
11	11010	3	1	0	4	1	8088.04	75.00	25.00	0.00
12	11011	3	1	0	4	6	8133.04	75.00	25.00	0.00
13	11012	0	0	5	5	4	81.26	0.00	0.00	100.00
14	11013	0	0	5	5	4	113.96	0.00	0.00	100.00
15	11014	0	1	5	6	4	138.45	0.00	16.67	83.33
16	11015	1	0	2	3	1	2500.97	33.33	0.00	66.67
17	11016	1	0	2	3	2	2332.28	33.33	0.00	66.67
18	11017	3	0	1	4	5	6434.31	75.00	0.00	25.00
19	11018	3	0	4	7	3	6533.28	42.86	0.00	57.14
20	11019	0	8	25	33	1	882.70	0.00	24.24	75.76
21	11020	1	0	1	2	1	2316.97	50.00	0.00	50.00
22	11021	1	0	2	3	1	2371.96	33.33	0.00	66.67
23	11022	1	0	1	2	2	2322.28	50.00	0.00	50.00
24	11023	0	1	5	6	2	122.24	0.00	16.67	83.33
25	11024	0	0	6	6	2	56.51	0.00	0.00	100.00
26	11025	3	2	1	6	4	6576.80	50.00	33.33	16.67
27	11026	3	1	3	7	3	6575.79	42.86	14.29	42.86
28	11027	3	2	4	9	4	6591.57	33.33	22.22	44.44
29	11028	3	0	2	5	4	6474.62	60.00	0.00	40.00
30	11029	3	1	3	7	4	6565.29	42.86	14.29	42.86
31	11030	3	1	0	4	4	6471.32	75.00	25.00	0.00
32	11031	3	0	3	6	6	6478.60	50.00	0.00	50.00
33	11032	3	1	6	10	6	6525.56	30.00	10.00	60.00
34	11033	3	1	3	7	6	6495.29	42.86	14.29	42.86
35	11034	3	1	0	4	6	6491.32	75.00	25.00	0.00
36	11035	3	0	2	5	4	6451.31	60.00	0.00	40.00
37	11036	1	0	1	2	2	2354.98	50.00	0.00	50.00
38	11037	1	0	6	7	1	2374.23	14.29	0.00	85.71