```sql
--*  BUSIT 103           Assignment   #9              DUE DATE :  Consult course calendar
/*
Name: Christopher Singleton
Class: BUSIT103 - Online
Instructor: Art Lovestedt
Date: 11/22/2014
*/
-- You are to develop SQL statements for each task listed.
-- You should type your SQL statements under each task.

/*     Submit your .sql file named with your last name, first name and assignment # (e.g.,
GriggsDebiAssignment9.sql).
       Submit your file to the instructor using through the course site.  */

--     Do not remove the USE statement.
USE AdventureWorksDW2012;

-- Note 1:  When the task does not specify sort order, it is your responsibility to order the
-- information so that is easy to interpret and add an alias to any columns without a name.
-- Note 2:  When asked to calculate an average or a count, for example, and then write a statement
-- using that value, be sure you are using the subquery and not hard coding the value.
-- Note 3:  The questions are numbered. 1.a., 1.b., 2.a., 2.b., etc., to remind you of the steps in
-- developing and testing your queries/subqueries. The first steps will not require subqueries
-- unless specified. The last step in every sequence will require a subquery, regardless of
-- whether the result can be created using another method, unless otherwise specified.

--1.   Read all of the requests for question 1 before beginning. Instructions in later requests
--            may answer questions about earlier requests. The joins are not complex but the WHERE is.

--1.a. (2) List the ProductKey, ProductAlternateKey, ProductSubcategoryKey, EnglishProductName,
--            FinishedGoodsFlag, Color, ListPrice, Size, Class, StartDate, EndDate, and Status for all
--            current products. One table only. Look at the results and pay attention to the values in
--            the fields. Understanding the data will help you make decisions about your filters in the
--            following statements. You will want to find simple filters that are sustainable--will still
--            work when the data set grows. Be sure to add a meaningful sort. Hint: Don't know how
--            to find current products? Run the statement with the WHERE and look for current.

SELECT [ProductKey]
      ,[ProductAlternateKey]
         ,[ProductSubcategoryKey]
         ,[EnglishProductName]
         ,[FinishedGoodsFlag]
         ,[Color]
         ,[ListPrice]
         ,[Size]
         ,[Class]
         ,[StartDate]
         ,[EndDate]
         ,[Status]
FROM [dbo].[DimProduct]
WHERE [Status] = 'Current'
ORDER BY [EnglishProductName]



-- 1.b. (1) List the distinct ProductKey for products sold to Resellers. (One table, one field, many rows)
--            No sort needed. Here you need to understand in which table sales to Resellers are stored.

SELECT DISTINCT [ProductKey]
FROM [dbo].[FactResellerSales]
```

```
-- 1.c. (3) Using an Outer Join find current Products that have not been sold to Resellers. Show Product
--            Key and the English Product Name. Add a meaningful sort.

SELECT p.[ProductKey] AS p_ProductKey --,frs.[ProductKey]  AS frs_ProductKey
      ,P.[EnglishProductName]
FROM [dbo].[DimProduct] AS p
     LEFT OUTER JOIN [dbo].[FactResellerSales] AS frs ON p.[ProductKey] = frs.[ProductKey]
WHERE frs.[ProductKey] IS NULL
ORDER BY P.[EnglishProductName]

--Art's NOTE-- 1c WHERE clause should also include P.Status = 'current' - 1d/1e


--1.d.  (2) Using the Outer Join from 1.c. find all current products have not been sold to Resellers
--            and are for sale (they are not inventory). Show Product Key, the English Product Name, and the
--            field(s) you used to find products that are for sale. Add a meaningful sort. Recall that
inventory
--            was talked about in Assignment 7, Question 2c. There are several ways to find products that are
--            for sale. Pick a method that works and makes sense to you. Include a comment about why
--            you chose the method you did.

SELECT p.[ProductKey] AS p_ProductKey
         ,p.[EnglishProductName]
         ,frs.[ProductKey] AS frs_ProductKey
         ,frs.[OrderQuantity] AS frs_OrderQuantity
         ,p.[Status] AS p_Status
FROM [dbo].[DimProduct] AS p
     LEFT OUTER JOIN [dbo].[FactResellerSales] AS frs ON p.[ProductKey] = frs.[ProductKey]
WHERE frs.[OrderQuantity] IS NULL AND p.[Status] IS NULL
ORDER BY p.[EnglishProductName]

/*
Art's Notes:- 1d/1e; no points off here
, but this should also include P.Status = 'current'in
the WHERE clause - 2b needs to INNER join to filter on
ProductSubcategoryKey: INNER JOIN [dbo].[DimProductSubcategory] AS PS
ON PS.ProductSubcategoryKey = P.ProductSubcategoryKey INNER JOIN [dbo].[DimProductCategory] AS PC
ON PC.ProductCategoryKey = PS.ProductCategoryKey WHERE PC.ProductCategoryKey = 4
*/
/*
Comment: I believe that OrderQuantity would be NULL if no products have sold
and also Status would be NULL if there are no products in inventory.
*/
```

```
--1.e.  (3) Rewrite the Outer Join from 1d as a subquery to find all current Products that are for sale and
--            have not been sold to Resellers. HINT: Review 1a and 1b. There will be no joins in the statement
for 1e.
--            1b will be used as a subquery in the WHERE clause to return a list. You want to find product
keys that
--            are not in that list and are for sale. This statement is likely simpler than you think it should
be.

SELECT p.[ProductKey] AS p_ProductKey
         ,p.[EnglishProductName]
         ,p.[Status] AS p_Status
FROM [dbo].[DimProduct] AS p
WHERE p.ProductKey NOT IN
                     (SELECT [ProductKey]
                                    FROM [dbo].[FactResellerSales] AS frs
                                    WHERE p.ProductKey = frs.ProductKey)
     AND p.[Status] IS NULL
ORDER BY p.[EnglishProductName]

/* Art's Notes:- 1d/1e; no points off here
, but this should also include P.Status = 'current'in
the WHERE clause - 2b needs to INNER join to filter on
ProductSubcategoryKey: INNER JOIN [dbo].[DimProductSubcategory] AS PS
ON PS.ProductSubcategoryKey = P.ProductSubcategoryKey INNER JOIN [dbo].[DimProductCategory] AS PC
ON PC.ProductCategoryKey = PS.ProductCategoryKey WHERE PC.ProductCategoryKey = 4
*/
--Example:
/*
SELECT c.CustFirstName, c.CustLastName, o.OrderNumber,
o.OrderDate, od.ProductNumber, p.ProductName, od.QuantityOrdered
FROM (
      (Customers AS c INNER JOIN Orders AS o ON c.CustomerID = o.CustomerID)
       INNER JOIN Order_Details od ON o.OrderNumber = od.OrderNumber
        )
INNER JOIN Products p ON p.ProductNumber = od.ProductNumber
WHERE o.OrderDate = (SELECT MAX(OrderDate) FROM Orders AS O2 WHERE O2.CustomerID = c.CustomerID);
*/


-- 2.a.       (4) List the average listprice of accessory items for sale by AdventureWorks. No sort
--            needed. Remember to provide a column alias. Use the AVG function that was demonstrated
--            in the Subqueries Demo file.

SELECT AVG(ListPrice) AS AvgListPrice
FROM [dbo].[DimProduct] AS p
WHERE ProductSubCategoryKey IN
                 (SELECT ProductSubCategoryKey
                       FROM [dbo].[DimProductSubcategory] AS psc
                       WHERE psc.ProductSubCategoryKey = p.ProductSubCategoryKey
                                AND ProductCategoryKey = 4)


/*SELECT AVG(YearlyIncome) AS AvgAnnualIncome
FROM dimCustomer
--WHERE EnglishEducation = 'graduate degree' --Used to spot check the results set*/
```

```sql
-- 2.b. (3) List the products in the Accessory category that have a listprice higher than the average
--            listprice of Accessory items.  Show product alternate key, product name, and listprice in the
--            results set. Order the information so it is easy to understand. Be sure
--            to use a subquery; do not enter the actual value from 2.a. into the statement.

SELECT dp.ProductAlternateKey
      ,dp.EnglishProductName
      ,dp.ListPrice
FROM [dbo].[DimProduct] AS dp
WHERE dp.ListPrice > (SELECT Avg(ListPrice)
                        FROM [dbo].[DimProduct] AS p
                           WHERE p.ProductSubCategoryKey IN
                                                                     (SELECT psc.ProductSubCategoryKey
                                                          FROM [dbo].[DimProductSubcategory] AS psc
                                                          WHERE psc.ProductSubCategoryKey =
p.ProductSubCategoryKey
                                                                   AND psc.ProductCategoryKey = 4))
ORDER BY dp.ListPrice

-- 3.a. (2) Find the average yearly income of all houseowners in the customer table.

SELECT Avg(YearlyIncome) AS AvgYearlyIncome
FROM [dbo].[DimCustomer]

/*
Art's Notes: 3a is missing: WHERE [HouseOwnerFlag] = 1 - 3b;
no points off here, but this also needs the HouseOwnerFlag filter.
*/
-- 3.b. (3) Find all houseowners in the customers table with an income less than or the same as
--            the average income of all customers. List last name, a comma and space, and first name in
--            one column, the customer key, and yearly income. There will be three columns in the Results
--            set. Be sure to use a subquery; do not enter the actual value from 3.a. into the statement.

SELECT LastName + ' ' + FirstName AS FullName
      ,CustomerKey
         ,YearlyIncome
FROM [dbo].[DimCustomer]
WHERE YearlyIncome <= (SELECT Avg(YearlyIncome) AS AvgYearlyIncome
                       FROM [dbo].[DimCustomer])
ORDER BY YearlyIncome

-- 4.a.       (2) List the product name and list price for the bike named Road-150 Red, 62

SELECT EnglishProductName
      ,ListPrice
FROM [dbo].[DimProduct]
WHERE EnglishProductName = 'Road-150 Red, 62'


-- 4.b.       (3) List the product name and price for each bike that has a price greater than or equal to
--         that of the Road-150 Red, 62. Be sure you are using the subquery not an actual value.

SELECT EnglishProductName
      ,ListPrice
FROM [dbo].[DimProduct]
WHERE ListPrice >= (SELECT ListPrice
                    FROM [dbo].[DimProduct]
                                 WHERE EnglishProductName = 'Road-150 Red, 62')
ORDER BY EnglishProductName
```

```
/*     Questions 5 and 6 ask you to experiment with a few of the Special Predicate Keywords for
       Subqueries where requested. There are other ways to solve the statements, but use the requested
       predicate for practice. */

-- 5.a.       (3) List the names of resellers and the product names of products they purchased.
--      Eliminate duplicate rows. Use an appropriate sort. No special predicated requested.

SELECT DISTINCT
       (SELECT ResellerName FROM [dbo].[DimReseller] AS r WHERE r.ResellerKey = frs.ResellerKey) AS
r_ResellerName
      ,(SELECT EnglishProductName FROM [dbo].[DimProduct] AS p WHERE p.ProductKey = frs.ProductKey) AS
p_ProductName
FROM [dbo].[FactResellerSales] AS frs
ORDER BY r_ResellerName, p_ProductName

-- 5.b.       (3) List the names of all resellers who sold a Road-150 Red, 62. Eliminate duplicate
--      rows. Use the IN predicate and a subquery to accomplish the task. Use an appropriate
--            sort. The WHERE clause in this one is similar to, but less complex than, the one in 1.e.

SELECT DISTINCT
       r.ResellerName
FROM [dbo].[DimReseller] AS r
WHERE r.ResellerKey IN (SELECT ResellerKey
                        FROM [dbo].[FactResellerSales] AS frs
                                          WHERE frs.ResellerKey = r.ResellerKey
                                                AND frs.ProductKey IN
                                                                     (SELECT ProductKey
                                                                      FROM [dbo].[DimProduct] AS p
                                                       WHERE p.ProductKey = frs.ProductKey
                                                                                            AND
p.EnglishProductName = 'Road-150 Red, 62'))
ORDER BY r.ResellerName
/*--Checking
SELECT DISTINCT
       r.ResellerKey
      ,r.ResellerName
      ,p.EnglishProductName
         ,p.ProductKey
FROM [dbo].[DimReseller] r
     INNER JOIN [dbo].[FactResellerSales] AS frs ON frs.ResellerKey = r.ResellerKey
        INNER JOIN [dbo].[DimProduct] AS p ON frs.ProductKey = p.ProductKey
WHERE p.EnglishProductName = 'Road-150 Red, 62'
*/


-- 6.a. (1) Show all data from the Survey Response fact table. Use select all. No special predicate.

SELECT *
FROM [dbo].[FactSurveyResponse]
ORDER BY EnglishProductSubcategoryName
```

```
-- 6.b. (4) Use a subquery and the EXISTS predicate to find customers that respond to surveys. List full
--           name (first, middle, last) and email address (2 columns). Use the CONCAT() function for the name
--           to overcome the NULL issue. You will not see NULL in any row. Refer to the selected solutions
demo
--           in the Module 03 discussion board for help with CONCAT. EXISTS is in the Module 09 demo file.
--           NOTE: Don't overuse EXISTS. It appears easy to use but it may not give the results expected.

/*My Note:
If concatenating a value with null, you should use ISNULL(ColumnName,'') function to replace null with
nothing.
Otherwise, the sql will show whole concatenation value as NULL.
*/
SELECT FirstName + ' ' + ISNULL(MiddleName,'') + ' ' + LastName AS FullName
      ,EmailAddress
FROM [dbo].[DimCustomer] AS c
WHERE EXISTS
           (SELECT *
                  FROM [dbo].[FactSurveyResponse] AS fsr
                     WHERE fsr.CustomerKey = c.CustomerKey)
ORDER BY FullName


-- 6.c. (2) Copy/paste 6.b and use an additional subquery in the WHERE clause in the outer
--           query to narrow the results of 6.b. to only those customers with a yearly income that
--           is greater than or the same as the average of all customers.

SELECT FirstName + ' ' + ISNULL(MiddleName,'') + ' ' + LastName AS FullName
      ,EmailAddress
FROM [dbo].[DimCustomer] AS c
WHERE EXISTS
           (SELECT *
                  FROM [dbo].[FactSurveyResponse] AS fsr
                     WHERE fsr.CustomerKey = c.CustomerKey)
         AND c.YearlyIncome >= (SELECT AVG(YearlyIncome)
                                 FROM [dbo].[DimCustomer])
ORDER BY FullName


-- 6.d. (1) Modify 6.c to find those customers at the income level specified that who do not respond
--           to surveys. This modification requires the addition of one operator.

SELECT FirstName + ' ' + ISNULL(MiddleName,'') + ' ' + LastName AS FullName
      ,EmailAddress
FROM [dbo].[DimCustomer] AS c
WHERE NOT EXISTS
           (SELECT *
                  FROM [dbo].[FactSurveyResponse] AS fsr
                     WHERE fsr.CustomerKey = c.CustomerKey)
         AND c.YearlyIncome >= (SELECT AVG(YearlyIncome)
                                 FROM [dbo].[DimCustomer])
ORDER BY FullName


-- 7.a.      (1) Find the average of total children for all customers.
--           Use the Average function and provide an appropriate alias.

SELECT AVG(TotalChildren) AS AvgTotalChildren
FROM [dbo].[DimCustomer]
```

```
-- 7.b.       (3) Use a correlated subquery to find customers who have more children than the
--            average for customers in their same occupation. List customer key, last name,
--            first name, total children, and English occupation. Add a meaningful sort.
--            In a correlated subquery the inner query is dependent on the outer query for its value.
--            There is an example of a similar request in the Subqueries demo file.
--Thinking in sets:
SELECT CustomerKey
      ,LastName
         ,FirstName
         ,TotalChildren
         ,EnglishOccupation
FROM [dbo].[DimCustomer] AS c1
WHERE c1.TotalChildren > (SELECT AVG(TotalChildren)   --This is a correlated subquery - Inner query
                          FROM [dbo].[DimCustomer] AS c2   --getting the Average Number of Childern in a same
Occupation as the outer query.
                                        WHERE c2.EnglishOccupation = c1.EnglishOccupation)
ORDER BY EnglishOccupation, TotalChildren DESC

/*
--Checking Average of each occupation's total children:
SELECT EnglishOccupation, AVG(TotalChildren) AS AVG_TotalChildren
FROM [dbo].[DimCustomer]
GROUP BY EnglishOccupation
ORDER BY AVG_TotalChildren
*/

-- 8.  (4) List resellers of any business type who have annual sales above the average
--            annual sales for resellers whose Business Type is "Warehouse". Show Business type,
--            Reseller Name, and annual sales. Use appropriate subqueries.

SELECT BusinessType
      ,ResellerName
         ,AnnualSales
FROM [dbo].[DimReseller]
WHERE AnnualSales > (SELECT AVG(AnnualSales)
                     FROM [dbo].[DimReseller]
                                    WHERE BusinessType = 'Warehouse')
ORDER BY ResellerName
/*
Excellent job on this Chris! A few notes: - 1c WHERE clause should also include
P.Status = 'current' - 1d/1e; no points off here
, but this should also include P.Status = 'current'in
the WHERE clause - 2b needs to INNER join to filter on
ProductSubcategoryKey: INNER JOIN [dbo].[DimProductSubcategory] AS PS
ON PS.ProductSubcategoryKey = P.ProductSubcategoryKey INNER JOIN [dbo].[DimProductCategory] AS PC
ON PC.ProductCategoryKey = PS.ProductCategoryKey WHERE PC.ProductCategoryKey = 4 AND . . . . -
3a is missing: WHERE [HouseOwnerFlag] = 1 - 3b;
no points off here, but this also needs the HouseOwnerFlag filter
Art Lovestedt, Nov 28 at 12:03pm
```