

```
--* BUSIT 103
    DUE DATE: 10/11/2014 11:59PM
```

Assignment #3

```
/*
Name: Christopher Singleton
Class: BUSIT103 - Online
Instructor: Art Lovestedt
Date: 10/10/2014
*/
```

```
/* You are to develop SQL statements for each task listed. You should type your SQL statements
    under each task. You should always create an alias for any derived fields. Add a sort that makes
    sense for each query. */
```

```
/* Submit your .sql file named with your last name and first name and assignment #
    (e.g., GriggsDebiAssignment3.sql). Submit your file to the instructor using
    through the course site. */
```

USE AdventureWorksLT2012;

```
--1. (4) List the customer id and the name for each customer using two columns. The customer id will
--      be in the first column. Create a concatenation for the second column that combines the
title,
--      first name, and last name for each customer. For example, the name for customer ID 29485
will
--      display in one column as
--              Ms. Catherine Abel
--      Don't forget to include a space between each part of the name. Assign CustomerName as the
alias
--      for the derived column. Order the results in alphabetical order by last name then by
first name.
```

```
SELECT [CustomerID], [Title] + ' ' + [FirstName] + ' ' + [LastName] AS CustomerName
FROM [SalesLT].[Customer]
ORDER BY [LastName], [FirstName]
```

```
--2. (3) Using the CAST function, list the customer ID and the name for each customer in one column.
--      Create a concatenation of the customer id, title, first name and last name for each
customer. For
--      example, the record for customer id 29485 will display in one column as
--              29485 Ms. Catherine Abel
--      Assign CustomerInfo as the alias for the derived column.
--      Order the results in alphabetical order by last name then by first name.
--      HINT: Look at the data type of the fields to which you are concatenating the customer id
and cast
--      customer id to match.
```

```
--CAST Syntax
--CAST(Column AS DataType)
/*SELECT LEN([CustomerID])
FROM [SalesLT].[Customer]
LEN is 5 Characters
Concatenation Syntax '+' +Column+' '+Column AS Alias */
```

```
SELECT CAST([CustomerID] AS NVARCHAR(5)) + ' ' + [Title] + ' ' + [FirstName] + ' ' + [LastName] AS
CustomerInfo
FROM [SalesLT].[Customer]
ORDER BY [LastName], [FirstName] ASC
```

```
--3. (3) Using the CAST function, rewrite the SELECT statement created in #2 to add the descriptive
text
--      "Customer ID" and "is". The record for customer id 29485 will display in one column as
--      Customer ID 29485 is Ms. Catherine Abel
--      Use the same alias and sort order as #2.
-- This is just adding the string 'Customer ID' before the CAST function. 'space'+ 'is'+ 'space'...
```

```
SELECT 'Customer ID ' +
CAST([CustomerID] AS NVARCHAR(5)) + ' ' + 'is' + ' ' + [Title] + ' ' + [FirstName] + ' ' + [LastName] AS
CustomerInfo
FROM [SalesLT].[Customer]
ORDER BY [LastName], [FirstName] ASC
```

```
--4. (4) Using the CAST function and the ProductCategory table, create a list of the product category
--      and the category name in one column. Product category 1 will display in one column as
--      Product Category 1: Bikes
--      Give the derived column a meaningful alias (column name) and sort order.
```

```
SELECT 'Product'+ ' ' + 'Category'+ ' ' +
CAST([ProductCategoryID] AS NVARCHAR(2))+ ':' + ' ' + [Name] AS ProductCategory
FROM [SalesLT].[ProductCategory]
ORDER BY [Name] ASC
```

```
--5. For a and b below, use the SalesLT.SalesOrderDetail table to list all product sales.
--      Show SalesOrderID, TotalCost and LineTotal for each sale. Compute TotalCost as
--      UnitPrice * (1-UnitPriceDiscount)* OrderQty. Display money values to exactly 2 decimal
places.
--      TotalCost and LineTotal should show the same amount. LineTotal is included to double
check
--      your calculation; the two amounts should match. Be sure to add a meaningful sort to the
statement.
```

```
--a. (5) CAST is the ANSI standard. Write the statement using CAST.
--      TotalCost and LineTotal should show the same amount. LineTotal is included to double
check
```

```
SELECT [SalesOrderID], [LineTotal],
CAST(UnitPrice * (1-UnitPriceDiscount)* OrderQty AS Decimal(7,2)) AS TotalCost
FROM [SalesLT].[SalesOrderDetail]
ORDER BY TotalCost DESC
```

```
--b. (4) Write the statement again using CONVERT instead of CAST. CONVERT is also commonly used.
--      CONVERT is demonstrated in the Module 03 materials.
```

```
--CONVERT SYNTAX
--CONVERT(datatype, ColumnName or Expression)
--CONVERT(datatype, ColumnName, style)
```

```
SELECT [SalesOrderID], [LineTotal],
CONVERT(Decimal(7,2), UnitPrice * (1-UnitPriceDiscount)* OrderQty) AS TotalCost
FROM [SalesLT].[SalesOrderDetail]
ORDER BY TotalCost DESC
```

```
--6. For a. and b. below, AdventureWorks predicts a 6% increase in production costs for all their
-- products. They wish to see how the increase will affect their profit margins. To help
them
-- understand the impact of this increase in production costs (StandardCost), you will
create
-- a list of all products showing ProductID, Name, ListPrice, FutureCost (use StandardCost *
1.06
-- to compute FutureCost), and Profit (use ListPrice minus the calculation for FutureCost to
find Profit).
-- All money values are to show exactly 2 decimal places. Order the results descending by
Profit.
-- Hint: See the Module 03 Discussion posting on Logical Order of Operations. It will
explain why you
-- cannot use an alias created in the SELECT clause in a calculation but can use it in the
ORDER BY clause.
```

```
-- a. (5) First write the requested statement using CAST. CAST is the ANSI standard. There will be
five
-- fields (columns). There will be one row for each product in the Product table.
```

```
SELECT [ProductID], [Name], [ListPrice], CAST([StandardCost]*1.06 AS Decimal(6,2)) AS FutureCost,
CAST([ListPrice] - ([StandardCost]*1.06) AS Decimal(6,2)) AS Profit
FROM [SalesLT].[Product]
ORDER BY Profit DESC
```

```
--b. (4) Next write the statement from 6a again using CONVERT. There will be five
-- fields (columns). There will be one row for each product in the Product table.
```

```
--CONVERT(Datatype(LEN), Expression)
SELECT [ProductID], [Name], [ListPrice],
CONVERT(Decimal(6,2), [StandardCost]*1.06) AS FutureCost,
CONVERT(Decimal(6,2), [ListPrice] - ([StandardCost]*1.06)) AS Profit
FROM [SalesLT].[Product]
ORDER BY Profit DESC
```

```
--7. For a. and b. below, list all sales orders showing PurchaseOrderNumber, SalesOrderID,
CustomerID, OrderDate,
-- DueDate, and ShipDate. Format the datetime fields so that no time is displayed. Be sure
to give each derived
-- column an alias and add a meaningful sort to each statement.
```

```
--a. (6) CAST is the ANSI standard. Write the statement using CAST.
```

```
--CAST(ColumnName AS Datatype(Length))
SELECT [PurchaseOrderNumber], [SalesOrderID], [CustomerID],
CAST([OrderDate] AS NVARCHAR(11)) AS OrderDate,
CAST([DueDate] AS NVARCHAR(11)) AS DueDate,
CAST([ShipDate] AS NVARCHAR(11)) AS ShipDate
FROM [SalesLT].[SalesOrderHeader]
ORDER BY [SalesOrderID] ASC
```

```
--b. (5) Write the statement again using CONVERT.
```

```
--CONVERT(Datatype(length), ColumnName, style)
SELECT [PurchaseOrderNumber], [SalesOrderID], [CustomerID],
CONVERT(NVARCHAR(10), [OrderDate], 101) AS OrderDate,
CONVERT(NVARCHAR(10), [DueDate], 101) AS DueDate,
CONVERT(NVARCHAR(10), [ShipDate], 101) AS ShipDate
FROM [SalesLT].[SalesOrderHeader]
ORDER BY [SalesOrderID] ASC
```

```
--c. (5) Write a statement using either 7a or 7b add a field that calculates the
--      difference between the due date and the ship date. Name the field ShipDays and show
--      the result as a positive number. Be sure Datetime fields still show only the date.
--      The DateDiff function is not an ANSI standard; don't use it in this statement.
--DATEDIFF(DATETIME, StartDate, EndDate) AS Alias
```

```
SELECT [PurchaseOrderNumber], [SalesOrderID], [CustomerID],
CONVERT(NVARCHAR(10), [OrderDate], 101) AS OrderDate,
CONVERT(NVARCHAR(10), [DueDate], 101) AS DueDate,
CONVERT(NVARCHAR(10), [ShipDate], 101) AS ShipDate,
CAST(CONVERT(NVARCHAR(10), [DueDate], 112) as INT) -
CAST(CONVERT(NVARCHAR(10), [ShipDate], 112) as INT) AS ShipDays
FROM [SalesLT].[SalesOrderHeader]
ORDER BY [SalesOrderID] ASC
```

```
--d. BONUS Challenge (+2): Rewrite the statement from 7c to use the DateDiff function to find the
--      difference between the OrderDate and the ShipDate. Again, show only the date in datetime
--      fields.
```

```
SELECT [PurchaseOrderNumber], [SalesOrderID], [CustomerID],
CONVERT(NVARCHAR(10), [OrderDate], 101) AS OrderDate,
CONVERT(NVARCHAR(10), [DueDate], 101) AS DueDate,
CONVERT(NVARCHAR(10), [ShipDate], 101) AS ShipDate,
DATEDIFF(DAY, [OrderDate], [ShipDate]) AS InProcessDays,
DATEDIFF(DAY, [ShipDate], [DueDate]) AS ShipDays
FROM [SalesLT].[SalesOrderHeader]
ORDER BY [SalesOrderID] ASC
```

```
--8. (2) EXPLORE: Explore statements ask you to research your text or the Web for an answer. They are
--      not
--      extra credit questions.
--      Find a date function that will return a datetime value that contains the date and time
--      from the computer
--      on which the instance of SQL Server is running (this means it shows the date and time of
--      the PC on which
--      the function is executed). The time zone offset is not included. Write the statement so
--      it will execute.
--      Format the result to show only the date portion of the field and give it the alias of
--      MyPCDate.
```

```
SELECT CONVERT(VARCHAR(10), SYSDATETIME(), 101) as MyPCDate
```

/\*Much shorter to write and a bit easier to read :-)

Another example:

```
SELECT 'Product'+ ' '+Category+' ' + CAST([ProductCategoryID] AS NVARCHAR(2))+':'+ ' ' + [Name] AS
ProductCategory
```

Could be:

```
SELECT 'Product Category ' + CAST([ProductCategoryID] AS NVARCHAR(2)) + ': ' + [Name] AS
ProductCategory
```

- #5 LineTotal should just be 2 decimal places -

#7c specifically says to not use the DATEDIFF function :-) - Bonus Challenge; +2, nice job!

Art Lovestedt, Oct 16 at 7:41am \*/