```
/*
Chris Singleton                                                    02/12/2017
PROG 140                Module 4 Assignment
POINTS 50               DUE DATE:  February 12

Develop a SQL statement for each task listed.  This exercise uses the Northwind
database.

Please type your SQL statement under each task below.  Ensure your statement
works by testing it prior to turning it in When writing answers for your select
statements please keep in mind that there is more than one way to write a query.
Please do your best to write a query that not only returns the information the
questions ask for, but to the best of your ability, in the best way possible to
suit their needs. For example, consider the best way to sort, to name columns,
etc.
These are things you must think about on the job!

Turn In:

For this exercise you will submit two separate WORD documents (instead of a .sql
file) in which you have copied and pasted your entire work from your .sql files
including all assignment questions and your SQL queries. The first file will have
the answers to Questions 1-5 and the second file will be a script as described in
Questions 6-10. The first document should contain your name at the top,
assignment title, the date and any difficulties encountered. The second will be
as described in Questions 6-10 below.

Submit your files together to the instructor using the Canvas Assignment tool.

*/
-- Tasks:

/*
1.  Declare two variables of type Float. Set their values to 12 and 78,
respectively.  Write a SQL statement to multiply these two values together and
then add them to the literal numeric value 80:  12 * 78 + 80.  Your result will
be 1016.  Write both a Print and a Select statement to display the result of your
calculation.
*/
-- write your sql statements here:
/*Please Note: Variables are pre-initialized and we are using order of precedence
               to get 1016 (no parenthesis in our calculation).

Notes: Keep in mind that SELECT can also initialize multiple variables with
       values and print multiple values, but SET cannot initialize multiple
       variable values together (only one at a time) nor can you print multiple
       values using print.
```

```
 Hence, is valid: SELECT @FirstNum = 12, @SecondNum = 78,
        is NOT valid: SET @FirstNum = 12, @SecondNum = 78.
 Note: PRINT @FirstNum, @SecondNum --This will give an error.
 (You can only print one value at a time).
 SELECT can also just print values directly, but not in the message pane.
 Example:
   DECLARE @FirstNum float = 12, @SecondNum float = 78
   SELECT @FirstNum AS Num1, @SecondNum AS Num2
  */
--====================== My Answer ======================--

-- Declare and initialize the variables:
DECLARE @FirstNum float = 12,
        @SecondNum float = 78

--SET @FirstNum = 12; -- If you don't initialize to begin with, you can do this.
--SET @SecondNum = 78;

--Select what will show as a result while changing the datatype(s) as needed.
SELECT 'Total: ' + CAST((@FirstNum * @SecondNum + 80) AS varchar) AS
TotalCalculation;
GO --Signals the end of the batch.
--Declare and initialize the variables for the next batch (Print statement).
DECLARE @FirstNum float = 12,
        @SecondNum float = 78

/*Note: Below submits to the message pane. Above Submits as a column with
        values in a varchar string format to the results pane.*/
PRINT 'Total: ' + CAST((@FirstNum * @SecondNum + 80) AS varchar);
GO --Signals the end of the batch.
/*
2.  Declare a table variable that has the same first four columns as the Orders
table, ie., OrderID, CustomerID, EmployeeID, OrderDate.  Name it "OrdersCopy".
Use a single Insert statement to load this table variable with all Orders that
were shipped by "Speedy Express". Write a select statement to print the contents
of this table variable.
*/
-- write your sql statements here:

--====================== My Answer ======================--

--DECLARE @OrderID int, @CustomerID int, @EmployeeID int, @OrderDate date,
DECLARE @OrdersCopy TABLE
(  OrderNumber int PRIMARY KEY identity(1,1) --Surogate Key
  ,OrderID int
  ,CustomerID nchar(5)
  ,EmployeeID int
  ,OrderDate date
)
```

```sql
--Populate the table from Orders:
INSERT INTO @OrdersCopy(OrderID, CustomerID, EmployeeID, OrderDate)
SELECT o.OrderID
      ,o.CustomerID
      ,o.EmployeeID
      ,CAST(o.OrderDate AS date) AS OrderDate
FROM Orders AS o
INNER JOIN Shippers AS s ON o.ShipVia = s.ShipperID
WHERE CompanyName = 'Speedy Express' --Filter
ORDER BY OrderDate DESC --Show newest orders first.

SELECT * FROM @OrdersCopy -- View the table contents.

--Checking... ShipperID = 1
/*
  SELECT * FROM Shippers
  WHERE CompanyName = 'Speedy Express'

  SELECT * FROM Orders WHERE ShipVia = 1 --249 Rows (Correct)

3.  What will be the result of this statement?
          Declare @x int = 12;
          Print 'Result ' + @x
*/
-- write your answer here:

--======================= My Answer ========================--

/* Error, because you need to convert the numeral into a character format.
   Msg 245, Level 16, State 1, Line 100
   Conversion failed when converting the varchar
   value 'Result ' to data type int. */

--Should be:
DECLARE @x int = 12; --Initialize and declare the variable.
PRINT 'Result ' + CAST(@x AS char(9)) --Print out, cast int. to char.
GO --Signals the end of the batch.
--Output: Result 12
/*
4.  Rewrite the code in Step #3 so that it still uses both a single Declare
    Statement and a single Print statement but has a correct result.
*/
-- write your sql statements here:
--======================= My Answer ========================--
DECLARE @x int = 12; --Initialize and declare the variable.
PRINT 'Result ' + CAST(@x AS char(9)) --Print out, cast int. to char.
GO --Signals the end of the batch.
```

```sql
/*
5.  Declare a string variable and set its value equal to "Tokyo Traders".
    Write a query that uses your variable and joins the Products and Suppliers
    table to retrieve all Products for Supplier "Tokyo Traders" but write it so
    that the query won't run if the supplier "Tokyo Traders" does not exist in
    the Suppliers table.
*/
-- write your sql statements here:

--====================== My Answer ======================--

--Declare and initialize the variable @Tokyo for 'Tokyo Traders'
DECLARE @Tokyo varchar(20) = 'Tokyo Traders';
/*
Create a true/false (IF EXISTS) statement using IF ELSE,
then query the tables to get the information using an inner join
and the declared variable if it is true. If not true, then print out
'Tokyo Traders does not exist.'

Used an inner join to join the tables to get the information.
Note: Tokyo Traders does exist on one line in Suppliers (True).
*/

IF EXISTS (SELECT * FROM Suppliers WHERE CompanyName = @Tokyo)
     BEGIN
          SELECT p.ProductID
                ,p.ProductName
          FROM [dbo].[Products] AS p
          INNER JOIN [dbo].[Suppliers] AS s ON p.SupplierID = s.SupplierID
          WHERE s.CompanyName = @Tokyo
          ORDER BY CompanyName
          --Note: You could use another IF statement here (Nested IF's).
     END;
--ELSE SELECT 'Tokyo Traders does not exist.'  --You could also do this also.
ELSE --IF EXISTS is false, then print the message.
     BEGIN
          PRINT 'Tokyo Traders does not exist.'
     END;
```

```
--=============== Testing for False ====================--
/*

Checking...
SELECT * FROM Suppliers --Total: 29 Rows (including Tokyo Traders)
Where CompanyName = 'Tokyo Traders' --Will only display one Row.
*/
--Testing... for results:

--Create the temporary table CopySuppliers.
DECLARE @CopySuppliers TABLE
( SupplierID int, CompanyName nvarchar(40))

--Populate the table from Suppliers table in Northwind DB.
INSERT INTO @CopySuppliers (SupplierID, CompanyName)

--Select the columns to populate from Suppliers.
SELECT SupplierID
      ,CompanyName
FROM [dbo].[Suppliers]
WHERE CompanyName <> 'Tokyo Traders'
--Note: Leave out Tokyo Traders from being inserted.

ORDER BY CompanyName --Sort by CompanyName.

--Declare the variable that will hold the string 'Tokyo Traders'.
DECLARE @Tokyo varchar(20) = 'Tokyo Traders';

--Evaluate True or False using an IF statement.
IF EXISTS (SELECT *
           FROM @CopySuppliers
           WHERE CompanyName = @Tokyo)

           --IF True, then do this. Of course, 'Tokyo Traders does not exist.'
     BEGIN --Signifies the Beginning of the script.
           SELECT SupplierID
                 ,CompanyName
           FROM @CopySuppliers
           ORDER BY CompanyName
     END; --Signifies the End of the script.
--IF EXISTS is false, then do this.
ELSE
     BEGIN
           PRINT 'Tokyo Traders does not exist.'
     END; --Signifies the End of the script.

--=============================================================--
```