Name: Christopher Singleton
Class: Programming 140
Assignment: Module1
Date: 01/15/2017

SELECTS AND JOINS (Module1Exercisev2.sql)

Turn In:

For this exercise you will submit one WORD documents (instead of a .sql file) in which you have copied and pasted
your entire work from your .sql file including all assignment questions and your SQL queries.
The document should contain your name at the top, assignment title, the date and any difficulties encountered.
Submit your file to the instructor using the Canvas Assignment tool.

```
Use Northwind
-- Tasks:

/*
1.  Retrieve the customers from Mexico, Madrid and Sao Paulo. The list should include only
the company name, company contact person, and phone. Rename the columns in the output
so that they are more readable to a business user.
*/
-- write your sql statement here:

-- Students: I will answer this first one for you as an example of how I would like you to answer questions in our assignments!  -- Mary
-- please notice that in the following answer:
-- 1) all of the columns are well-named using aliasing
-- 2) the query uses an Order by clause to sort the result set
-- 3) the query itself is easy to read because it is nicely formatted rather than all on one line and messy :-)

SELECT c.CompanyName AS [Company]
, c.ContactName AS [Contact Name]
, c.Phone AS [Phone Number]
FROM Customers c
WHERE c.Country = 'Mexico' or c.City = 'Sao Paulo' or c.City = 'Madrid'
ORDER BY c.CompanyName

-- it is also important to note that the customer mixed both cities and countries in his/her request.  It is up to the
-- developer (you) to know the data well enough to give the user the correct result set!  Now the rest of the answers are up to you!
```

```
/*
2.  A business user from a team you support asks you:

"Can you please get me a list of our current Products that are priced between 20
and 25 dollars?
I need to know how many of each of these are in stock and how many are on order.
"

Write two queries to perform this.  One using a range query with comparison
operators and the other using "between"

My Notes: When we say a number between two numbers, the beginning point and end
point is not included. A number between 1 and 10 means (2, 3, 4, 5, 6, 7, 8, and
9).

*/
-- write your sql statements here:

SELECT ProductName
      ,UnitsInStock
      ,UnitsOnOrder
      ,UnitPrice
FROM Products
WHERE UnitPrice >= 21 AND UnitPrice <= 24
ORDER BY UnitPrice DESC


SELECT ProductName
      ,UnitsInStock
      ,UnitsOnOrder
      ,UnitPrice
FROM Products
WHERE UnitPrice BETWEEN 21 AND 24 --Between also includes beginning and end
points
ORDER BY UnitPrice DESC --, but the meaning in words is without beginning/end
points.

/*
3.
a)
Create a table called Recycle in the Northwind database using the following
criteria for the columns:

RecycleID: primary key, integer,
RecycleType: nchar(10), nulls not allowed
RecycleDescription nvarchar(100), null allowed
```

```
b)
Write insert statements so that the table is loaded with the following values:

RecycleID:  1
RecycleType: Compost
RecycleDescription:  Product is compostable, instructions included in packaging

RecycleID: 2
RecycleType:  Return
RecycleDescription:  Product is returnable to company for 100% reuse

RecycleID: 3
RecycleType: Scrap
RecycleDescription:  Product is returnable and will be reclaimed and reprocessed

RecycleID: 4
RecycleType: None
RecycleDescription: Product is not recycleable

c) Ensure that you have inserted data correctly by writing the SQL statement to
retrieve all data (all columns and all rows) from this table.
d) Write a statement to remove (drop) the table from the database
*/
-- write your sql statements here:
USE Northwind;
CREATE TABLE dbo.Recycle
    (RecycleID integer PRIMARY KEY,
      RecycleType nchar(10) NOT NULL,
      RecycleDescription nvarchar(100) NOT NULL)
GO

INSERT INTO Recycle
    (RecycleID, RecycleType, RecycleDescription)
Values(1,'Compost', 'Product is compostable, instructions included in packaging')

INSERT INTO Recycle
    (RecycleID, RecycleType, RecycleDescription)
Values(2,'Return', 'Product is returnable to company for 100% reuse')

INSERT INTO Recycle
    (RecycleID, RecycleType, RecycleDescription)
Values(3,'Scrap', 'Product is returnable and will be reclaimed and reprocessed')

INSERT INTO Recycle
    (RecycleID, RecycleType, RecycleDescription)
Values(4,'None', 'Product is not recycleable')

GO
--Checking...
SELECT TOP 4 * FROM dbo.Recycle
GO

DROP TABLE [dbo].[Recycle]
```

```
-----------------------------------------------------
-- Joins and Unions
-----------------------------------------------------
/*
4.  Starting with the query from question 1 add the OrderDate and ShippedDate
columns from the Orders table (Note: this will require a two table join. Also
don't be surprised that you will have more rows in the result set now because you
will have added information for the Customers orders!)
*/
-- write your sql statement here:

SELECT c.CompanyName AS [Company]
      ,c.ContactName AS [Contact Name]
      ,c.Phone AS [Phone Number]
      ,o.OrderDate AS [Order Date]
      ,o.ShippedDate AS [Shipped Date]
FROM Customers AS c
           INNER JOIN Orders AS o ON c.CustomerID = o.CustomerID
WHERE c.Country = 'Mexico' or c.City = 'Sao Paulo' or c.City = 'Madrid'
ORDER BY c.CompanyName

/*
5.  Starting with the query from question 4 add the Shippers ID and Shippers name
columns from the Shippers table (Note: this will NOT add more rows to your result
set).
*/
-- write your sql statement here:

SELECT c.CompanyName AS [Company]
      ,c.ContactName AS [Contact Name]
      ,c.Phone AS [Phone Number]
     ,o.OrderDate AS [Order Date]
     ,o.ShippedDate AS [Shipped Date]
     ,s.ShipperID AS [Shipper ID]
     ,s.CompanyName AS [Company Name]
FROM Customers AS c
           INNER JOIN Orders AS o ON c.CustomerID = o.CustomerID
           INNER JOIN Shippers AS s ON s.ShipperID = o.ShipVia
WHERE c.Country = 'Mexico' or c.City = 'Sao Paulo' or c.City = 'Madrid'
ORDER BY c.CompanyName
```

```sql
/*
6.  Starting with the query from question 1 write a query that counts the total
     Orders for each company.  You will need only the columns from question 1.
      (Hint: This will require a two table join. You will use the Count
aggregate, and you will group by company name, contactname and phone.)
*/
-- write your sql statement here:

SELECT COUNT(c.CompanyName) AS [Company Orders]
            ,c.ContactName AS [Contact Name]
            ,c.Phone AS [Phone Number]
FROM Customers AS c
                INNER JOIN Orders AS o ON c.CustomerID = o.CustomerID
WHERE c.Country = 'Mexico' or c.City = 'Sao Paulo' or c.City = 'Madrid'
GROUP BY c.CompanyName, c.ContactName, c.Phone
ORDER BY [Company Orders] DESC


/*
7.  Your boss asks you if we have any customers that have not ever
actually purchased anything as yet and if so to give her a list of them.
Please write a query to answer this question.
(Hint: This will be a two-table Left join!!)
*/
-- write your sql statement here:
/*Note: I would think they would want contact info (mailing, phone, fax, etc...)
        I could have did a select *, but this made more sense with performance
and checking.*/

SELECT c.CompanyName
      ,c.ContactName
      ,c.ContactTitle
      ,c.[Address] --Address is a keyword.
      ,c.City
      ,c.PostalCode
      ,c.Country
      ,c.Phone
      ,c.Fax
      --,o.OrderDate --Checking.
      --,o.OrderID   --Checking.
FROM Customers AS c
          LEFT JOIN Orders AS o ON c.CustomerID = o.CustomerID
WHERE o.OrderID IS NULL OR o.OrderDate IS NULL
ORDER BY CompanyName
```

```
/*
8.  Your team needs a phone list of Shippers and Suppliers. Write a single
query for this. Include only the Shipper and Supplier IDs and their names and
phone numbers
Therefore, your list will have a total of 3 columns. (Hint:  Union!)
*/
-- write your sql statement here:
SELECT ShipperID AS ShipperID_SupplierID, CompanyName, Phone
FROM Shippers
UNION
SELECT SupplierID, CompanyName, Phone --Second select columns are ignored.
FROM Suppliers
ORDER BY CompanyName


/*
9.  Consider the following SQL statement:

select * from SalesDetail;

Assume this statement is syntactically correct and that there is a database and
table that this will run correctly against. Assume this database is for a very
successful store with hundreds of thousands of daily sales.  Do you have any
thoughts or issues with running this query regularly?

*/
-- write your answer here (please write in a comment delimited by /* and */):

/* Yes, there are issues of performance when it comes to the select-all wildcard
(expecially when you have a large amount of rows). If at all possible, you should
use SELECT TOP 10 rows specific columns based on how many rows you will need to
analyze. This will help reduce performance bog down and allow the users to carry
on their transactions without very little delay. (always give a range of what you
selecting if at all possible) otherwise you can interrupt the user's daily
transactions happening in the database.
*/
```

```
/*
10.  Consider the following SQL statement:

SELECT VendorName, AccountDescription, COUNT(*) AS LineItemCount,
       SUM(InvoiceLineItemAmount) AS LineItemSum
FROM Vendors JOIN Invoices
   ON Vendors.VendorID = Invoices.VendorID
 JOIN InvoiceLineItems
   ON Invoices.InvoiceID = InvoiceLineItems.InvoiceID
 JOIN GLAccounts
   ON InvoiceLineItems.AccountNo = GLAccounts.AccountNo
GROUP BY VendorName, AccountDescription
ORDER BY VendorName, AccountDescription;

Please do your best explain in your own words what this query does. Explain as
though you are talking to a co-worker who is a non-technical member of your team.
You DO NOT need to run this query! Try to explain by analyzing the sql code
alone.

*/
-- write your answer here (please write in a comment delimited by /*  and */):

/* This query gives the co-worker the total number of invoice line items (each
product or service), the total line item amount for each vendor with the vendor
name and account description on each record.


First Were joining four tables with two matching columns (VendorName and
AccountDescription) in order to bring our information together that is in the two
specified columns from the select statement. The matching columns we are getting
are Vendor's name, Account Description while grouping, then counting all the line
items and adding the line items together in coming up with a sum of all line
items while giving the two aggregated result sets columns names (alias').

Finally, were sorting our column VendorName's information first in ascending
order and then the AccountDescription column second in ascending order. Note
ascending order is automatically made by default when not specified with ASC.*/
```

```
/* Joins */
Use Northwind
/*
Question from a user:
Please give me a list of Shippers and the orders they have shipped. Include the
OrderID and the date the order was shipped.
*/
-- first look at the ERD!
select top 5 * from shippers
select top 5 * from Orders

--================ My Answer ========================
--Note: Nulls indicate that the order has not shipped.
SELECT s.ShipperID
      ,s.CompanyName
       ,o.OrderID
       ,o.ShippedDate
FROM Shippers AS s
     INNER JOIN Orders o ON s.ShipperID = o.ShipVia
WHERE ShippedDate IS NOT NULL
ORDER BY ShippedDate DESC
--====================================================
--====================================================

-- what is the join column? One approach is to start with the join then select
the columns
--================ My Answer ========================
/*The INNER JOiN is created on the Primary/Foreign Keys (Shippers ShipperID PK
and Orders ShipVia FK). The first query selects allt he columns and will issue
all the rows in both tables being joined. The second query selects only the
columns listed in the select clause while joining Note: ShipVia is the foreign
key in the Orders table.*/

select *
from Shippers s join Orders o on s.ShipperID = O.ShipVia

select s.ShipperID, S.CompanyName, O.OrderID, O.ShippedDate as [Shipped Date]
from Shippers s join Orders o on s.ShipperID = O.ShipVia

--====================================================
--====================================================
-- 3 table join
/*  Thanks for the list you gave me earlier -- now can you tell me which employee
sold that order? */

select s.ShipperID, s.CompanyName, o.OrderID, o.ShippedDate,
            e.EmployeeID, e.Lastname
from Shippers s
     join Orders o on s.ShipperID = o.ShipVia
     join Employees e on e.EmployeeID = o.EmployeeID
Order by ShipperID, ShippedDate desc
```

```sql
--================ My Answer ========================
/*  Thanks for the list you gave me earlier -- now can you tell me which employee
sold that order? */
/*I provided the newest orders using the top 30.
  Note: Nulls mean they haven't shipped yet.
        Highest OrderID number is the newest order.*/
SELECT TOP 30 s.ShipperID
             ,s.CompanyName
             ,o.OrderID
             ,o.ShippedDate
             ,e.EmployeeID
             ,e.Lastname
FROM Shippers s
     INNER JOIN Orders o ON s.ShipperID = o.ShipVia
     INNER JOIN Employees e ON e.EmployeeID = o.EmployeeID
ORDER BY ShippedDate DESC, OrderID, ShipperID
--Make's more sense to order by shippedDate.

--=====================================================
--=====================================================
-- 4 table join
/* Now add the customer! */
select s.ShipperID, s.CompanyName, o.OrderID, o.ShippedDate,
            e.EmployeeID, e.Lastname, o.CustomerID, c.CompanyName
from Shippers s
     join Orders o on s.ShipperID = o.ShipVia
     join Employees e on e.EmployeeID = o.EmployeeID
     join Customers c on c.Customerid = o.Customerid
Order by ShipperID, ShippedDate desc
--================ My Answer ========================
-- 4 table join
/* Now add the customer! */
/*My guess would be that if they wanted the customer included and
that they would also want the contact info.*/
SELECT TOP 30 s.ShipperID
             ,s.CompanyName
                 ,o.OrderID
                 ,o.ShippedDate
                 ,e.EmployeeID
                 ,e.Lastname
                 ,c.ContactTitle
                 ,c.ContactName
                 ,c.[Address] --Note: Address is a key word.
                 ,c.City
                 ,c.Region
                 ,c.Phone
                 ,c.Fax
FROM Shippers s
     INNER JOIN Orders o ON s.ShipperID = o.ShipVia
     INNER JOIN Employees e ON e.EmployeeID = o.EmployeeID
     INNER JOIN Customers c ON o.CustomerID = c.CustomerID
ORDER BY OrderID DESC, ShippedDate, ShipperID
```

```sql
--=====================================================
--=====================================================
-- "Now also please tell me how many orders each Shipper has shipped for us"
select count(*) from shippers

select Shipperid, orderid
from  Shippers s join Orders o on s.ShipperID = O.ShipVia
order by ShipperID

select s.ShipperID
, S.CompanyName
, count(*) as TotalOrders
from Shippers s join Orders o on s.ShipperID = O.ShipVia
group by s.ShipperID, S.CompanyName
order by totalorders desc
--================ My Answer ========================
-- "Now also please tell me how many orders each Shipper has shipped for us"
/*Note: Nulls indicate that the shipper hasn't shipped yet. */
SELECT s.ShipperID
      ,s.CompanyName
      ,COUNT(*) AS TotalOrders
FROM Shippers s
      INNER JOIN Orders o ON s.ShipperID = O.ShipVia
WHERE o.ShippedDate IS NOT NULL --Filtered the NULL's out.
GROUP BY s.ShipperID, S.CompanyName
ORDER BY totalorders DESC
```

RESULTS:

ShipperID	CompanyName	TotalOrders
2	United Package	315
3	Federal Shipping	249
1	Speedy Express	245

```sql
--=====================================================
--=====================================================
-- Outer joins
-- join the two tables Employees to Orders
select *
from Employees e join Orders o on e.EmployeeID = o.Employeeid
order by e.Employeeid

-- Question:  How many employees have never sold anything? One (Obama).
Insert into Employees (Lastname, FirstName) values ('Obama', 'Barack')

select e.Lastname, o.Orderid
from Employees e left join Orders o on e.EmployeeID = o.EmployeeID
where o.OrderID is null

-- this query is functionally equivalent to the one above
select e.Lastname, o.Orderid
from Orders o right join Employees e  on e.EmployeeID = o.EmployeeID
where o.OrderID is null
```

```sql
--================ My Answer ========================
--Obama, Barack is NULL on the OrderID, so that means he never sold anything.
--===================================================

-- what a dead beat!!

/* Unions */
-- let's use the union to create a contacts list of both employees and customer
contacts
-- simple union:
select contactname as Contact, Phone, Country
from Customers
union
select Lastname, HomePhone, Country
from Employees

-- what are some problems with this result set?
-- let's clean it up:
select contactname as Contact, Phone as [Home Phone], Country
from Customers
union -- automatically eliminates dups
select FirstName + ' ' + Lastname, HomePhone, Country
from Employees
Order by Country, Contact

select contactname as Contact, Phone as [Home Phone], Country
from Customers
union all --  leaves in dups
select FirstName + ' ' + Lastname, HomePhone, Country
from Employees
Order by Country, Contact
--================ My Answer ========================
-- what are some problems with this result set?
/*Should always filter by range to reduce the load on the database.
  Capitalize key words and re-organize the columns in the select statement (make
it easier to read). Don't concatenate First and Last names (they should be in
separate columns to begin with). If you find a need to concatenate, then use an
alias for column name. Be more specific of where your information is coming from
by using an alias for the tables (Customer AS c, Employees AS e). I don't see any
need for duplicates, because there are no duplicates (No need for JOIN ALL).
  Order by ContactName makes more sense than by country. */

SELECT c.contactname AS Contact
      ,c.Phone AS [Home Phone]
       ,c.Country
FROM Customers AS c
UNION
SELECT e.FirstName + ' ' + e.Lastname AS ContactName
      ,e.HomePhone
       ,e.Country
FROM Employees AS e
ORDER BY ContactName, Country
--===================================================
```