Lab 2-2

Connection values:

Server Type = Database Engine
Server Name = boyce.coe.neu.edu
Authentication = SQL Server Authentication
Login = INFO6210
Password = NEUHusky!

Note:
Two ways to specify comments in SQL commands:
Use -- for a line of comments
or use /* */ for a block of comments.

```sql
-- Set the database context
USE AdventureWorks2008R2;
-- Or any version of AdventureWorks after it

-- SQL JOINs are used to retrieve data from multiple tables.
-- INNER is the default when JOIN is the only keyword used.
-- INNER JOIN returns only matching rows from left and right tables.
-- c is the alias for the Sales.Customer table in the example.
-- oh is the alias for the Sales.SalesOrderHeader table.
-- ON lists the matching columns to JOIN on.

/*
   If two tables have the same column name in a query, we must
   designate where the column is from by using the format
   TableName.ColumnName.
   If a column name is unique between the JOINed tables,
   The TableName.ColumnName format is not required.
*/

SELECT c.CustomerID, c.AccountNumber, SalesOrderID, OrderDate
FROM Sales.Customer c
INNER JOIN Sales.SalesOrderHeader oh
ON c.CustomerID = oh.CustomerID;

/*
   LEFT OUTER JOIN returns all rows from the left table,
   but only the matching rows from the right table.
*/

SELECT c.CustomerID, c.AccountNumber, SalesOrderID, OrderDate
FROM Sales.Customer c
LEFT OUTER JOIN Sales.SalesOrderHeader oh
ON c.CustomerID = oh.CustomerID;

/*
   RIGHT OUTER JOIN returns all rows from the right table,
   but only the matching rows from the left table.
*/

SELECT c.CustomerID, c.AccountNumber, SalesOrderID, OrderDate
FROM Sales.Customer c
RIGHT OUTER JOIN Sales.SalesOrderHeader oh
ON c.CustomerID = oh.CustomerID;
```

```sql
-- Set the database context
USE AdventureWorks2008R2;

-- COUNT, GROUP BY, ORDER
-- GROUP BY aggregates on the column(s) we specify
-- ORDER BY does sorting


SELECT c.CustomerID,
       PersonID,
       COUNT(SalesOrderID) AS "Total Order"
FROM Sales.Customer c
INNER JOIN Sales.SalesOrderHeader oh
ON c.CustomerID = oh.CustomerID
GROUP BY c.CustomerID, PersonID
ORDER BY "Total Order" DESC;
```

```
--JOIN, COUNT, GROUP BY, HAVING, ORDER

--SELECT the order count for each customer
--WHERE the count > 20
--ORDER the counts in the descending order

/*
For regular filtering in a query, we use WHERE.
If we use GROUP BY in a query, then we use HAVING to do
the filtering for groups.
*/

SELECT c.CustomerID,
       PersonID,
       COUNT(SalesOrderID) AS "Total Order"
FROM Sales.Customer c INNER JOIN Sales.SalesOrderHeader oh
ON c.CustomerID = oh.CustomerID
GROUP BY c.CustomerID, PersonID
HAVING COUNT(SalesOrderID) > 20
ORDER BY "Total Order" DESC;
```

	CustomerID	PersonID	Total Order
1	11091	4515	28
2	11176	15994	28
3	11185	12569	27
4	11200	5409	27
5	11223	3197	27
6	11262	20532	27
7	11276	15449	27
8	11277	4855	27
9	11287	15978	27
10	11300	13098	27

```sql
-- Set the database context
USE AdventureWorks2008R2;

-- IN OPERATOR

-- Can be used with any data type

SELECT ProductID, Name, Color, ListPrice, SellStartDate
FROM Production.Product
WHERE Color IN ('Red', 'Blue', 'White') -- character comparison
ORDER BY Color, Name;

SELECT ProductID, Name, Color, ListPrice, SellStartDate
FROM Production.Product
WHERE ListPrice IN (337.22, 594.83, 63.50, 8.99) -- numeric comparison
ORDER BY ListPrice;


-- LIKE operator
-- Select any person whose last name begins with a
-- % is the wildcard symbol representing 0 to many characters
-- - is the wildcard symbol representing exactly one character

SELECT FirstName, MiddleName, LastName
FROM Person.Person
WHERE LastName LIKE 'a%'
ORDER BY LastName;

-- Select any person whose last name begins with a or c or e

SELECT FirstName, MiddleName, LastName
FROM Person.Person
WHERE LastName LIKE '[ace]%'
ORDER BY LastName;
```

```sql
-- Set the database context
USE AdventureWorks2008R2;


--Subqueries are queries that are embedded in another query.

SELECT Name [Product],
   ListPrice,
   (SELECT MAX(ListPrice) FROM Production.Product)
           AS [Max Price],
   (ListPrice / (SELECT MAX(ListPrice) FROM Production.Product)) * 100
           AS [Percent of MAX]
FROM Production.Product
WHERE ListPrice > 0
ORDER BY ListPrice DESC;
```

```
-- Lab 2 Questions


Note: 1 point for each question

/* Use the content of the AdventureWorks2008R2 database for each of
   the following questions. Submit the SQL queries to Canvas in
   a single .sql file. */


-- 2-1

/*      Retrieve the salesperson ID, the most recent order date
        and the total number of orders processed by each salesperson
        for each salesperson. Use a column alias to make the report
        more presentable if a column heading is missing. Use CAST to
        display only the date of the order date. Exclude the orders
        which don't have a salesperson specified.

        Sort the returned data by the total number of orders in
        descending.

        Hints: (a) You need to work with the Sales.SalesOrderHeader table.

               (b) The syntax for CAST is CAST(expression AS data_type),
                where expression is the column name we want to format and
                we can use DATE as data_type for this question to display
                just the date.  */


--2-2

/* Write a query to select the product id, name, and list price
   for the product(s) that have a list price greater than the
   average list price plus $500.

   Use a column alias to make the report more presentable
   if a column heading is missing. Sort the returned data by the
   list price in descending.

   Hint: You'll need to use a simple subquery to get the average
         list price and use it in a WHERE clause. */



-- 2-3

/* Write a query to calculate the "orders to customer ratio"
   (number of orders / unique customers) for each sales territory.

   Return only the sales territories which have a ratio >= 5.
   Include the Territory ID and Territory Name in the returned data.
   Sort the returned data by TerritoryID. */
```

```
2-4
/* Write a query to retrieve the total sold quantity for each product.
   Return only the products that have a total sold quantity greater than 3000
   and have the black color.

   Use a column alias to make the report look more presentable if a column
   heading is missing. Sort the returned data by the total sold quantity
   in the descending order. Include the product ID, product name and
   total sold quantity columns in the report.

   Hint: Use the Sales.SalesOrderDetail and Production.Product tables. */




-- 2-5

/* Write a query to retrieve the dates in which
   there was at least one order placed but no order
   worth more than $500 was placed. Use TotalDue
   in Sales.SalesOrderHeader as the order value.

   Return the "order date" and "total product quantity sold
   for the date" columns. The order quantity column is
   in SalesOrderDetail. Display only the date part of the
   order date.

   Sort the returned data by the
   "total product quantity sold for the date" column in desc. */




-- 2-6

/*
Write a query to return the year and total sales of orders
on the new year day for each year. Please keep in mind
the database has several years' data.

Include only orders which contained 42 or more unique products
when calculating the total sales.

Use TotalDue in SalesOrderHeader as an order's value when calculating
the total sales. Return the total sales as an integer. Sort the
returned data by year.
*/
```

Useful Links

USE SQL Server Management Studio

http://msdn.microsoft.com/en-us/library/ms174173.aspx

Writing SQL Queries

http://technet.microsoft.com/en-us/library/bb264565(v=sql.90).aspx

SQL Aggregate Functions

http://msdn.microsoft.com/en-us/library/ms173454.aspx

Types of JOIN in SQL Server

http://www.codeproject.com/Tips/712941/Types-of-Join-in-SQL-Server

GROUP BY and HAVING
http://technet.microsoft.com/en-us/library/ms180199.aspx

Subquery Fundamentals

http://technet.microsoft.com/en-us/library/ms189575(v=sql.105).aspx

CAST and CONVERT

https://msdn.microsoft.com/en-us/library/ms187928.aspx