Lab 3

Connection values:

Server Type = Database Engine
Server Name = is-swang01.ischool.uw.edu
Authentication = SQL Server Authentication
Login = INFO6210
Password = NEUHusky!

```sql
/* CASE function allows conditional processing. */

-- Example of a CASE function
-- The ROUND function does number rounding

USE AdventureWorks2008R2;

SELECT
      ProductID
      , Name
      , ListPrice
      , (SELECT ROUND(AVG(ListPrice), 2) AS AvgPrice
        FROM Production.Product) AP
      , CASE
           WHEN ListPrice - (SELECT ROUND(AVG(ListPrice), 2)
                AS AvgPrice FROM Production.Product) = 0
                THEN 'Average Price'
           WHEN ListPrice - (SELECT ROUND(AVG(ListPrice), 2)
                AS AvgPrice FROM Production.Product) < 0
                THEN 'Below Average Price'
           ELSE 'Above Average Price'
        END AS PriceComparison
FROM Production.Product
ORDER BY ListPrice DESC;
```

```
/*
   Use the RANK function without/with the PARTITION BY clause
   to return the rank of each row.
*/



-- Without PARTITION BY

/*
   If the PARTITIAN BY clause is not used, the entire row set
   returned by a query will be treated as a single big partition.
*/

USE AdventureWorks2008R2;

SELECT
      RANK() OVER (ORDER BY OrderQty DESC) as [Rank],
      SalesOrderID, ProductID, UnitPrice, OrderQty
FROM Sales.SalesOrderDetail
WHERE UnitPrice >75;



-- With PARTITION BY

/*
   When the PARTITIAN BY clause is used, the ranking will be
   performed within each partitioning value.
*/


SELECT
      RANK() OVER (PARTITION BY ProductID ORDER BY OrderQty DESC)
        as [Rank],
      SalesOrderID, ProductID, UnitPrice, OrderQty
FROM Sales.SalesOrderDetail
WHERE UnitPrice >75;
```

```
-- RANK

/*
If two or more rows tie for a rank, each tied row receives the same
rank. For example, if the two top salespeople have the same SalesYTD
value, they are both ranked one. The salesperson with the next highest
SalesYTD is ranked number three, because there are two rows that are
ranked higher. Therefore, the RANK function does not always return
consecutive integers. Sometimes we say the RANK function creates gaps.
*/
```

```
-- DENSE_RANK

/*

If two or more rows tie for a rank in the same partition, each tied
row receives the same rank. For example, if the two top salespeople
have the same SalesYTD value, they are both ranked one. The
salesperson with the next highest SalesYTD is ranked number two. This
is one more than the number of distinct rows that come before this
row. Therefore, the numbers returned by the DENSE_RANK function do not
have gaps and always have consecutive ranks.

*/

USE AdventureWorks2008R2;
GO
SELECT i.ProductID, p.Name, i.LocationID, i.Quantity
    ,DENSE_RANK() OVER
    (PARTITION BY i.LocationID ORDER BY i.Quantity DESC) AS Rank
FROM Production.ProductInventory AS i
INNER JOIN Production.Product AS p
    ON i.ProductID = p.ProductID
WHERE i.LocationID BETWEEN 3 AND 4
ORDER BY i.LocationID;
GO
```

Here is the result set.

```
ProductID   Name                               LocationID Quantity Rank
----------- ---------------------------------- ---------- -------- ---
494         Paint - Silver                     3          49       1
495         Paint - Blue                       3          49       1
493         Paint - Red                        3          41       2
496         Paint - Yellow                     3          30       3
492         Paint - Black                      3          17       4
495         Paint - Blue                       4          35       1
496         Paint - Yellow                     4          25       2
493         Paint - Red                        4          24       3
492         Paint - Black                      4          14       4
494         Paint - Silver                     4          12       5

(10 row(s) affected)
```

-- Lab 3 Questions

```
Note: 1.2 points for each question
Use the content of the AdventureWorks sample database.

Lab 3-1
/* Modify the following query to add a column that identifies the
   frequency of repeat customers and contains the following values
   based on the number of orders during 2007:
     'No Order' for count = 0
     'One Time' for count = 1
     'Regular' for count range of 2-5
     'Often' for count range of 6-10
     'Loyal' for count greater than 10

   Give the new column an alias to make the report more readable.
*/

SELECT c.CustomerID, c.TerritoryID,
COUNT(o.SalesOrderid) [Total Orders]
FROM Sales.Customer c
LEFT OUTER JOIN Sales.SalesOrderHeader o
   ON c.CustomerID = o.CustomerID
WHERE DATEPART(year, OrderDate) = 2007
GROUP BY c.TerritoryID, c.CustomerID;

Lab 3-2
/* Modify the following query to add a rank with gaps in the ranking
   based on the total orders in the descending order. Also partition
   by territory.

   Give the new column an alias to make the report more readable. */

SELECT c.CustomerID, c.TerritoryID,
COUNT(o.SalesOrderid) [Total Orders]
FROM Sales.Customer c
LEFT OUTER JOIN Sales.SalesOrderHeader o
ON c.CustomerID = o.CustomerID
WHERE DATEPART(year, OrderDate) = 2007
GROUP BY c.TerritoryID, c.CustomerID;
```

```
Lab 3-3
/* Write a query that returns the highest bonus amount received for
   the male sales people in North America. */


Lab 3-4
/* Retrieve the product id of the top selling product of each day.
   Use the total sold quantity to determine the top selling product.
*/


Lab 3-5
/* Provide a unique list of customer id’s and account numbers which
   have ordered both products 711 and 712 after July 1, 2008.
   Sort the list by customer id. */
```

Useful Links

SQL CASE Functions

http://msdn.microsoft.com/en-us/library/ms181765.aspx

SQL Ranking Functions

http://msdn.microsoft.com/en-us/library/ms189798.aspx

SQL DATEPART Function

http://msdn.microsoft.com/en-us/library/ms174420.aspx