

## Lab 3

Connection values:

Server Type = Database Engine

Server Name = boyce.coe.neu.edu

Authentication = SQL Server Authentication

Login = INF06210

Password = NEUHusky!

```
/* 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 PARTITION 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 PARTITION 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.

\*/

/\*

RANK() creates GAPS (missing numbers).

DENSE\_RANK() does not create GAPS.

\*/

SELECT

RANK() OVER (ORDER BY OrderQty DESC) AS [Rank],

SalesOrderID, ProductID, UnitPrice, OrderQty

FROM Sales.SalesOrderDetail

WHERE UnitPrice >75;

Rank	SalesOrderID	ProductID	UnitPrice	OrderQty
1	53460	976	850.495	30
2	55282	954	1192.035	26
3	71783	976	850.495	25
4	51131	892	552.1505	23
4	47395	760	430.6445	23
6	51132	973	935.5445	22

```
-- 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 AdventureWorks2008R2 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:

'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, FirstName, LastName,
COUNT(o.SalesOrderid) [Total Orders]
FROM Sales.Customer c
JOIN Sales.SalesOrderHeader o
    ON c.CustomerID = o.CustomerID
JOIN Person.Person p
    ON p.BusinessEntityID = c.PersonID
WHERE c.CustomerID > 25000
GROUP BY c.TerritoryID, c.CustomerID, FirstName, LastName;
```

-- Lab 3-2

/\* Modify the following query to add a rank without gaps in the ranking based on total orders in the descending order. Also partition by territory.\*/

```
SELECT c.CustomerID, c.TerritoryID, FirstName, LastName,
COUNT(o.SalesOrderid) [Total Orders]
FROM Sales.Customer c
JOIN Sales.SalesOrderHeader o
    ON c.CustomerID = o.CustomerID
JOIN Person.Person p
    ON p.BusinessEntityID = c.PersonID
WHERE c.CustomerID > 25000
GROUP BY c.TerritoryID, c.CustomerID, FirstName, LastName;
```

-- Lab 3-3

/\* Retrieve the date, product id, product name, and the total sold quantity of the worst selling (by total quantity sold) product of each date. If there is a tie for a date, it needs to be retrieved.

Sort the returned data by date in descending. \*/

-- Lab 3-4

/\* Write a query to retrieve the most valuable salesperson of each year.  
The most valuable salesperson for each year is the salesperson who has  
made most sales for AdventureWorks in the year.

Calculate the yearly total of the TotalDue column of SalesOrderHeader  
as the yearly total sales for each salesperson. If there is a tie  
for the most valuable salesperson, your solution should retrieve it.  
Exclude the orders which didn't have a salesperson specified.

Include the salesperson id, the bonus the salesperson earned,  
and the most valuable salesperson's total sales for the year  
columns in the report. Display the total sales as an integer.  
Sort the returned data by the year. \*/

-- Lab 3-5

/\*  
Write a query to return the salesperson id, the most sold product id,  
and the order id that contained the highest total order quantity for  
each salesperson. The most sold product had the highest total order quantity.

Return only the salesperson(s) who had at least one order that contained  
a total sold quantity greater than 450. Exclude orders which don't have  
a salesperson for this query. Sort the returned data by the salesperson 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>