

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 AdventureWorks sample database.

Lab 3-1

/* Modify the following query to add a column that identifies the performance of salespersons and contains the following feedback based on the number of orders processed by a salesperson:

'Need to Work Hard' for the order count range 1-100
'Fine' for the order count range of 101-300
'Strong Performer' for the order count greater than 300

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

```
SELECT SalesPersonID, p.LastName, p.FirstName,  
       COUNT(o.SalesOrderid) [Total Orders]  
FROM Sales.SalesOrderHeader o  
JOIN Person.Person p  
  ON o.SalesPersonID = p.BusinessEntityID  
GROUP BY o.SalesPersonID, p.LastName, p.FirstName  
ORDER BY p.LastName, p.FirstName;
```

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 o.TerritoryID, s.Name, year(o.OrderDate) Year,  
       COUNT(o.SalesOrderid) [Total Orders]  
FROM Sales.SalesTerritory s  
JOIN Sales.SalesOrderHeader o  
  ON s.TerritoryID = o.TerritoryID  
GROUP BY o.TerritoryID, s.Name, year(o.OrderDate)  
ORDER BY o.TerritoryID;
```

Lab 3-3

/* Write a query that returns the male salesperson(s) who received the lowest bonus amount in Europe. Include the salesperson's id and bonus amount in the returned data. Your solution must retrieve the tie if there is a tie. */

Lab 3-4

```
/* Write a query to retrieve the most valuable customer of each year.  
The most valuable customer of a year is the customer who has  
made the most purchase for the year. Use the yearly sum of the  
TotalDue column in SalesOrderHeader as a customer's total purchase  
for a year. If there is a tie for the most valuable customer,  
your solution should retrieve it.
```

```
Include the customer's id, total purchase, and total order count  
for the year. Display the total purchase in two decimal places.  
Sort the returned data by the year. */
```

Lab 3-5

```
/* Write a query to retrieve the dates in which there was  
at least one product sold but no product in red  
was sold.
```

```
Return the "date" and "total product quantity sold  
for the date" columns. The order quantity can be found in  
SalesOrderDetail. Display only the date for a date.
```

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

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>