Departement Informatika
Department of Informatics

MS SQL 2008

Advanced data retrieval

Chapter 9 (Part 1) | Hotek, 2008

Learning objectives

- Admin
- Aggregating data using the GROUP BY clause

Aggregate Functions

- Aggregate functions perform a calculation on a set of values and return a single value
- Except for COUNT, aggregate functions ignore null values.
- Aggregate functions are frequently used with the GROUP BY clause of the SELECT statement

Aggregate Functions

- Transact-SQL provides the following aggregate functions
  - COUNT - Returns the number of items in a group
  - MIN - Returns the minimum value in the expression
  - MAX - Returns the maximum value in the expression
  - AVG - Returns the average of the values in a group
  - SUM - Returns the sum of all the values, or only the DISTINCT values, in the expression. SUM can be used with numeric columns only.

The GROUP BY clause

- Groups a selected set of rows into a set of summary rows by the values of one or more columns or expressions.
- One row is returned for each group.
- Aggregate functions in the SELECT clause <select> list provide information about each group instead of individual rows.

GROUP BY clause

- The following query returns the number of Males and Females

```
USE AdventureWorks
GO

SELECT MaritalStatus, COUNT(*) AS Total
FROM HumanResources.Employee
GROUP BY MaritalStatus
ORDER BY MaritalStatus DESC
```

NB: All columns in the SELECT clause that are not aggregates must be included in the GROUP By clause

ORDER BY clause guarantees an order to be used

NULL values in GROUP BY with COUNT

- NULL values will also be grouped . For example

```
USE AdventureWorks
GO

SELECT Color, COUNT(*) AS 'Total'
FROM Production.Product
GROUP BY Color
```

Results | Messages

	Color	Total number
1	NULL	248
2	Black	93
3	Blue	26
4	Grey	1
5	Multi	8
6	Red	38
7	Silver	43
8	Silver/Black	7
9	White	4
10	Yellow	36

Using GROUP BY with multiple groups

- More than one group can be defined. For example

```
Use AdventureWorks

SELECT ProductID,
SpecialOfferID, AVG(UnitPrice)
AS 'Average Price',
SUM(LineTotal) AS SubTotal
FROM Sales.SalesOrderDetail
GROUP BY ProductID,
SpecialOfferID
ORDER BY ProductID
GO
```

Results | Messages

	ProductID	SpecialOfferID	Average Price	SubTotal
1	707	8	16.8221	2452.662180
2	707	11	15.7455	2971.175850
3	707	3	18.9272	2191.058910
4	707	1	31.3436	141271.252000
5	707	2	20.0556	8886.245452
6	708	3	18.9753	3461.676690
7	708	8	16.8221	2316.403170
8	708	11	15.7455	2997.943200
9	708	2	20.0502	11689.730276
10	708	1	30.9648	140403.764500
11	709	2	5.51	723.573200
12	709	3	5.225	853.765000
13	709	1	5.70	4235.100000

Aggregating multiple permutations

- Aggregate functions are more to be expected in the organisational setting

- For example, when the boss needs to know which Sales Representative has the greatest number of sales

```
USE AdventureWorks
Go

SELECT SalesPersonID,
SUM(TotalDue) Total
FROM Sales.SalesOrderHeader
GROUP BY SalesPersonID
ORDER BY Total DESC
GO
```

GROUP BY with multiple tables

For example in finding the number of employees for each City

```
USE AdventureWorks
GO
SELECT a.City, COUNT(ea.AddressID) EmployeeCount
FROM HumanResources.EmployeeAddress ea
   INNER JOIN Person.Address a
      ON ea.AddressID = a.AddressID
GROUP BY a.City
ORDER BY a.City
GO
```

Makes use of the JOIN operator

Practical exercise

- Run the SQL script for the MusicologyWarehouse
- Create 1 GROUP BY query to find out the total that each customer has spent.
- Will need to use JOINs and SUM clause

Results | Messages

	Customer_FullName	Total Spend
1	Wilmar Van Heerden	1368.00
2	Mike Theodorou	1309.00
3	Suzaan Coetzee	595.00
4	Vincenzo Bellomo	512.00
5	Lazo Karas	411.00
6	Andrew Suddards	355.00
7	Nsoozi Matongo	355.00
8	Stella Van Rensburg	256.00
9	Liza Kriek	250.00
10	Dario Macagnano	198.00