

6

Grouping Data Using Aggregate Functions

This chapter will cover the use of aggregate functions to group data to assist managers in the decision-making process. This type of grouping of data falls under the category of business intelligence where the processed data assists managers in performing strategic decision making. Often managers and business people need answers to such questions as those listed below:

- What region has the most employees?
- What is the average salary of all departments?
- What were the total salaries for a specific year?
- What is the percentage of female employees by department?

Answers to questions such as those above help businesses in the decision-making process to strategize growth and meet regulations. Aggregate functions provide the ability to answer these questions and much more.

CHAPTER OBJECTIVES

The learning objectives for this chapter are as follows:

- Learn to use the SUM, AVG, COUNT, MAX, and MIN functions in your queries.
- Learn how to use the GROUP BY clause to aggregate data by columns depending on needed information.
- Learn how filter aggregated data using the HAVING clause.

Using Aggregate Functions

Aggregate functions provide the ability to group data in multiple ways to answer questions, which lead to more knowledge-based decision making. Some of the most common aggregate functions are listed below:

- COUNT() – Retrieves the number of rows in a set
- MIN() – Retrieves the minimum value for a chosen column
- MAX() – Retrieves the maximum value for a chosen column
- AVG() – Retrieves the average value for a chosen column of numbers
- SUM() – Retrieves the sum value for a chosen column of numbers

COUNT() Function

The COUNT() function is used to count the number of rows in a set. For example, if a manager wanted to know the number of employees she had working for the company the SELECT statement in Figure 1 would provide the answer. Notice the alias “Total Employees” is provided as the column name.

```
SELECT COUNT (*) 'Total Employees'
FROM Employees
```

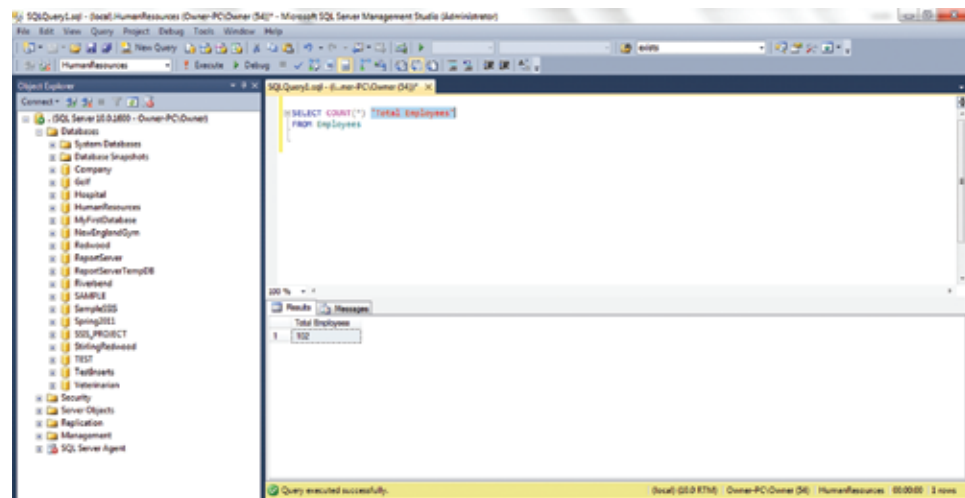


Figure 1

Notice the asterisk inside the parenthesis. The count function requires an argument to work. Arguments are also called parameters. The COUNT() function argument in Figure 1 is the asterisk. You can also use a column name for an argument. Figure 2 shows the same SELECT statement only in place of the asterisk the “FirstName” column is used.

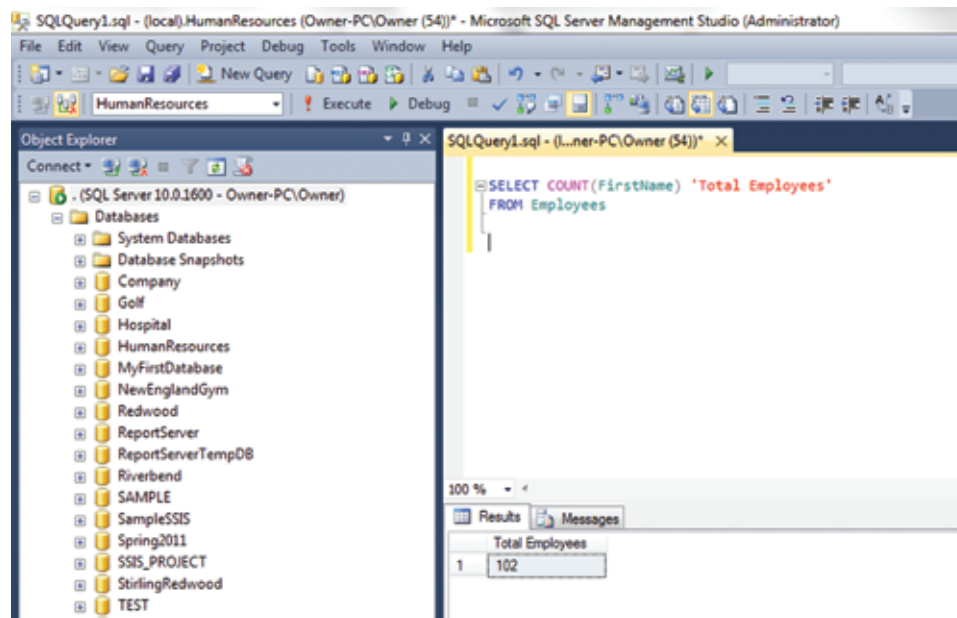


Figure 2

It is important to be aware that when column names are used null values will not be considered in the total. For example, the SELECT statement in Figure 3 is using the COUNT() function with the “MiddleName” column as the parameter. Since four employees are missing middle names the total Employees returned is 98, which is four less than our actual total of 102.

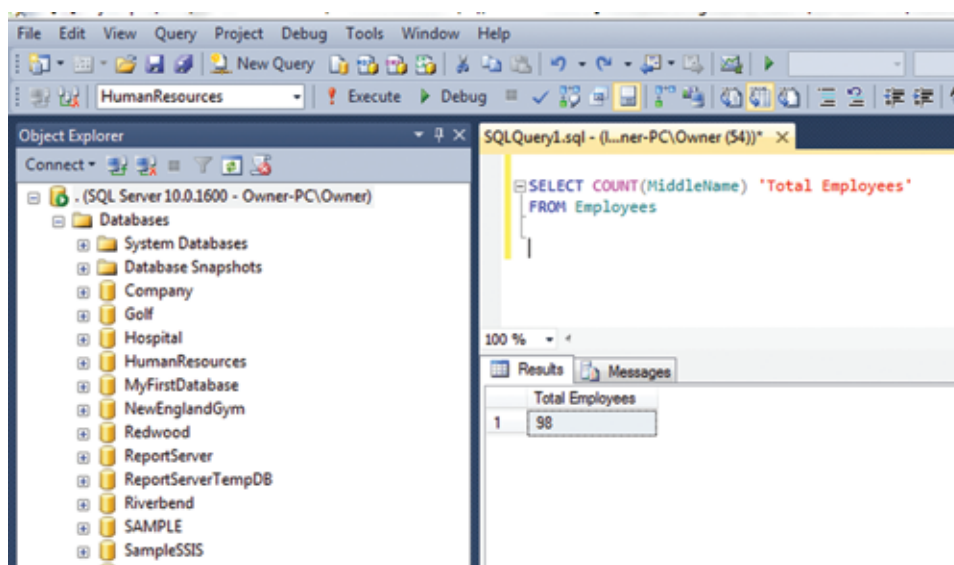


Figure 3

You can also use multiple COUNT() functions in a SELECT statement. Figure 4 shows an example where two COUNT() functions are being used with different arguments. The COUNT() function that is using “FirstName” as an argument returns a total of 102 while the COUNT() function that uses “MiddleName” as an argument returns only 98 as a total.

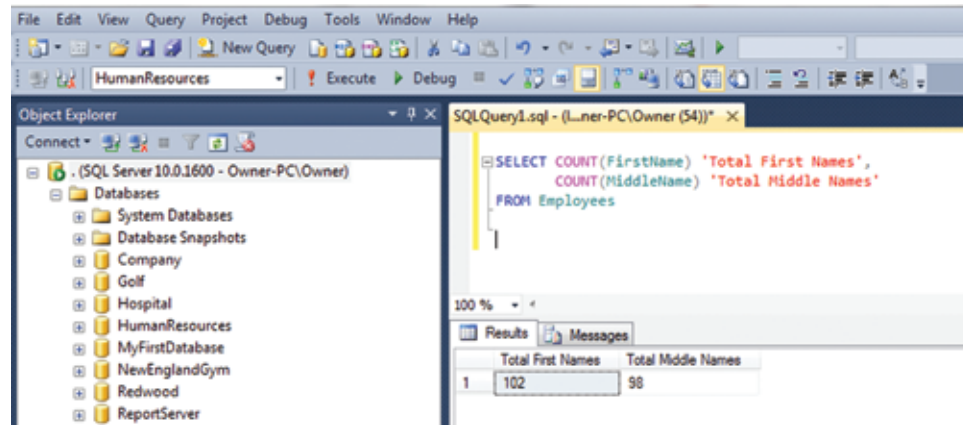


Figure 4

GROUP BY Clause

A useful way to use the COUNT() function is to use it to group by sets. For example, to find out how many employees exist for each title the SELECT statement can GROUP BY title as shown in Figure 5.

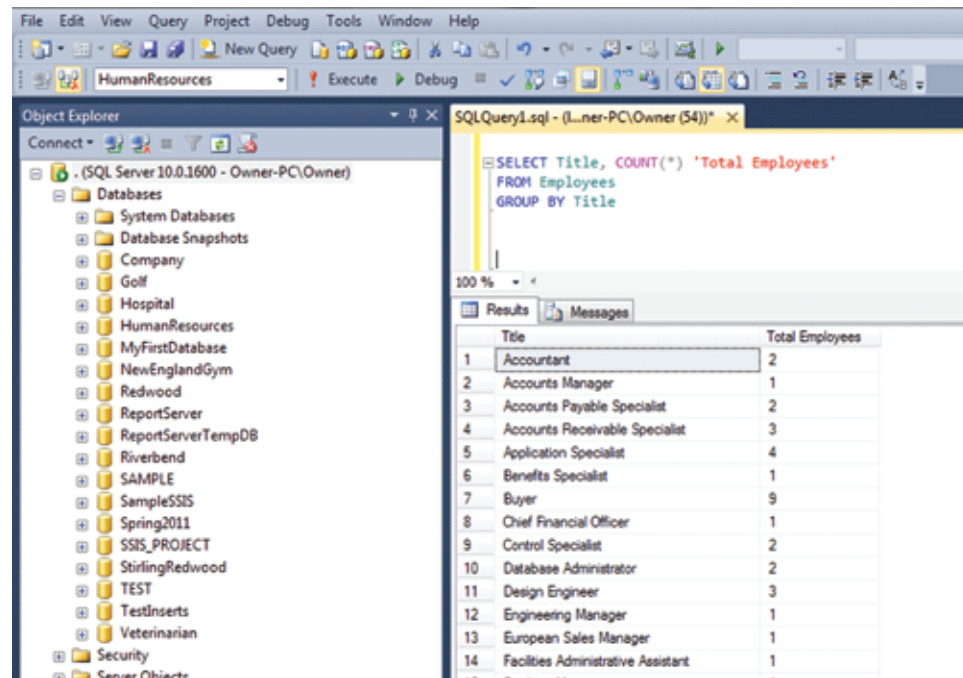


Figure 5

Notice the column “Title” is used following the GROUP BY clause because it is the column we are aggregating on. We can also group by multiple columns. Figure 6 shows grouping by “Title” and then “Gender” to show how many females and males belong to each title. Although I chose to order by “Title” and then “Gender” it wasn’t necessary, ordering by “Gender” and then “Title” would simply invert the column placement.

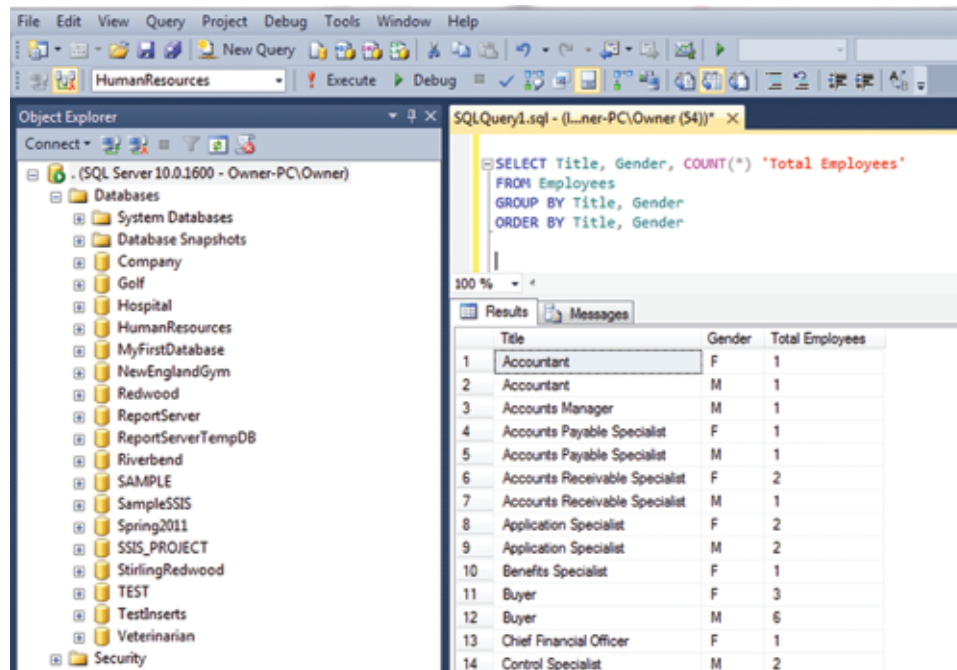


Figure 6

Errors

A common mistake made when using aggregate functions is to leave a column out of the GROUP BY clause when aggregating on it. For example, Figure 7 shows a SELECT statement where “Gender” is missing from the GROUP BY clause. If the column is not an argument for a function, then it must be in the GROUP BY clause or removed altogether. The resulting error is shown below.

Msg 8120, Level 16, State 1, Line 2

Column “Employees.Gender” is invalid in the select list because it is not contained in either an aggregate function or the GROUP BY clause.

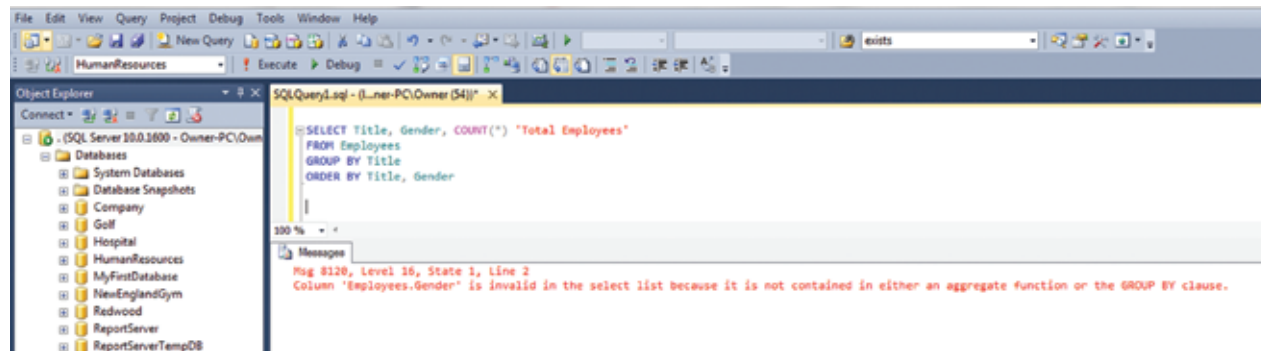


Figure 7

HAVING Clause

In place of the WHERE clause the HAVING clause is used to filter the result set for aggregate functions. Like the WHERE clause, the HAVING clause filters rows as well. The difference between the two is when the filtering occurs. The WHERE clause filters rows before the grouping action (before the aggregate calculation) and the HAVING clause filters rows after the grouping action (after the calculation action).

The SELECT statement example in Figure 8 is used to find the total number of male employees within each title where the count exceeds 2. The first step in this process is to remove any female employees and this is performed by the WHERE clause. After the grouping by title is performed the totals that are less than or equal to 2 are removed. Therefore, the results show total counts for titles that are associated with male employees and have counts greater than 2 for each title.

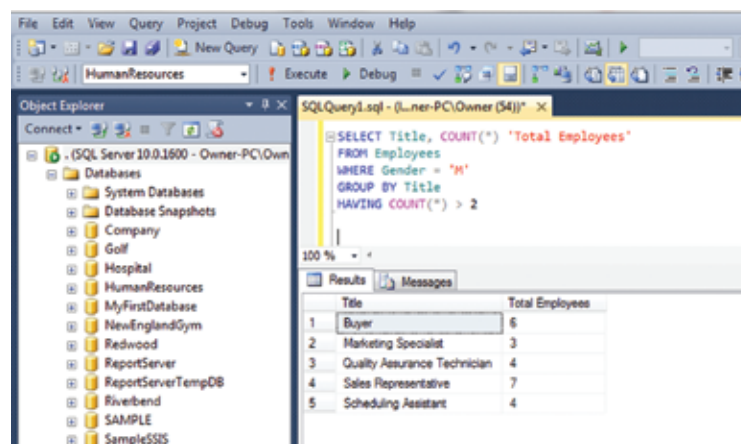


Figure 8

Remember that aggregate functions require the HAVING clause to filter rows. Below are two sample SELECT statements attempting to filter the data rows returned. The first example is incorrect because it is using the WHERE clause on an aggregate function. The second example is correct because it is properly using the HAVING clause.

This will not work:

```
SELECT Title, COUNT (*) 'Total Employees'
FROM Employees
GROUP BY Title
WHERE COUNT (*) > 4
```

This will work:

```
SELECT Title, COUNT (*) 'Total Employees'
FROM Employees
GROUP BY Title
HAVING COUNT (*) > 4
```

Basic Grouping Rules

The purpose of the GROUP BY clause is to have result sets that contain one row per group with one or more aggregate values. The following criteria must be applied to your queries to have them work properly.

- If column names that are not used as arguments in aggregate function are to be shown in the result set it is necessary to have them listed in the GROUP BY clause as well.
- Column names listed in the GROUP BY clause should also be listed in the SELECT clause.
- The HAVING clause cannot be used without the GROUP BY clause.
- Aggregate functions cannot be used with a WHERE clause.

AVG() Function

The average function returns the average value of a column within a set. For example, if the manager wanted to see the average salary for his company, the SELECT statement in Figure 9 would display that.

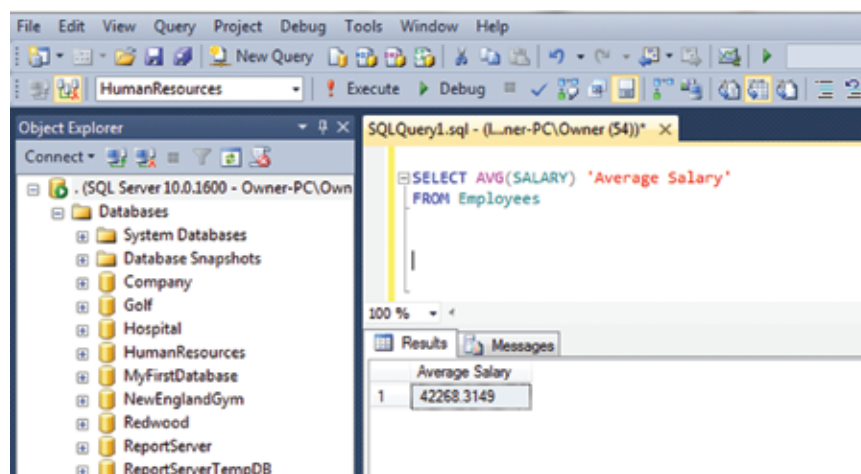


Figure 9

► **NOTE:** You will see results that contain more than two decimal points. This is a result from the way the database calculates the values. In a later chapter, we will learn how to format to two decimal places.

If a manager would like to see the average salaries for each title, the SELECT statement in Figure 10 would work.

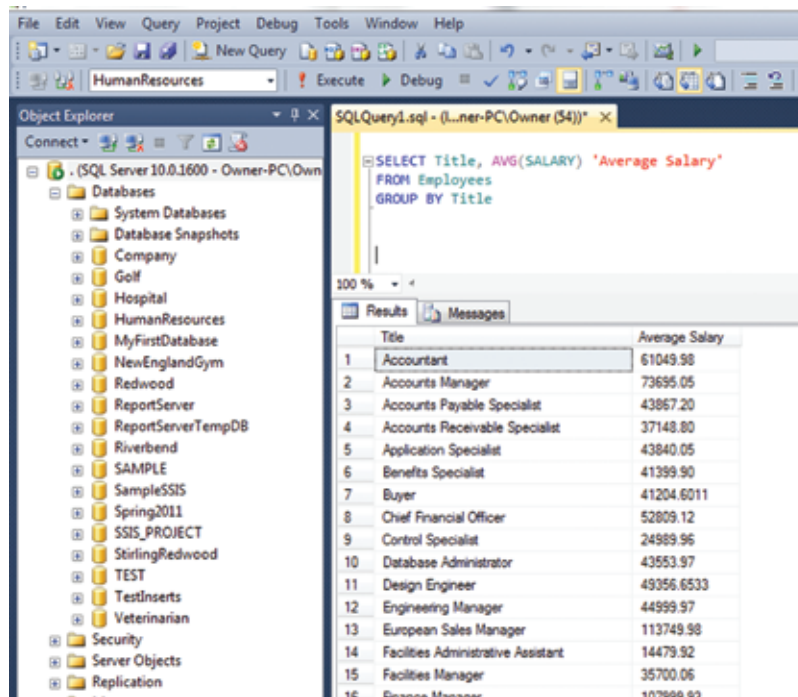


Figure 10

What if the manager would like to see the average salary by gender for employees who have the title “buyer”? The SELECT statement in Figure 11 would produce those results.

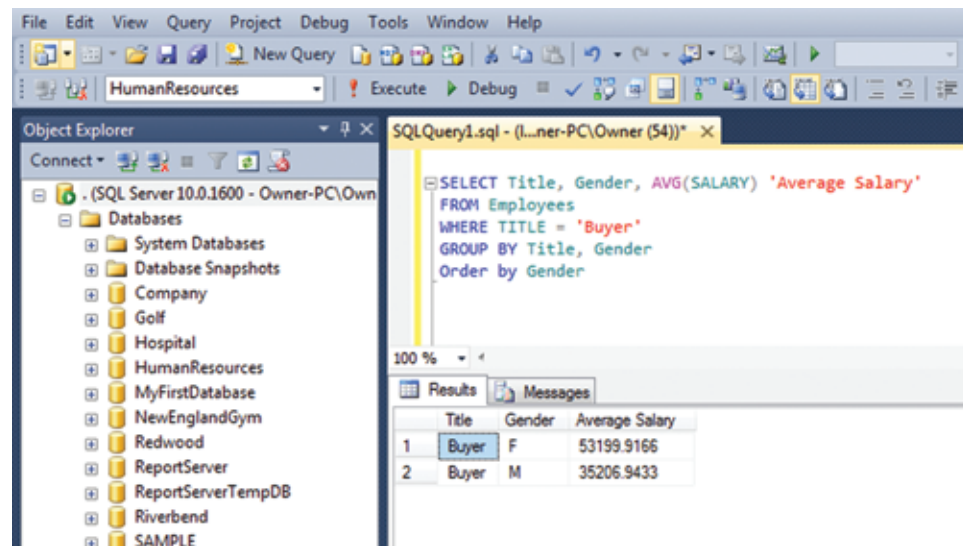
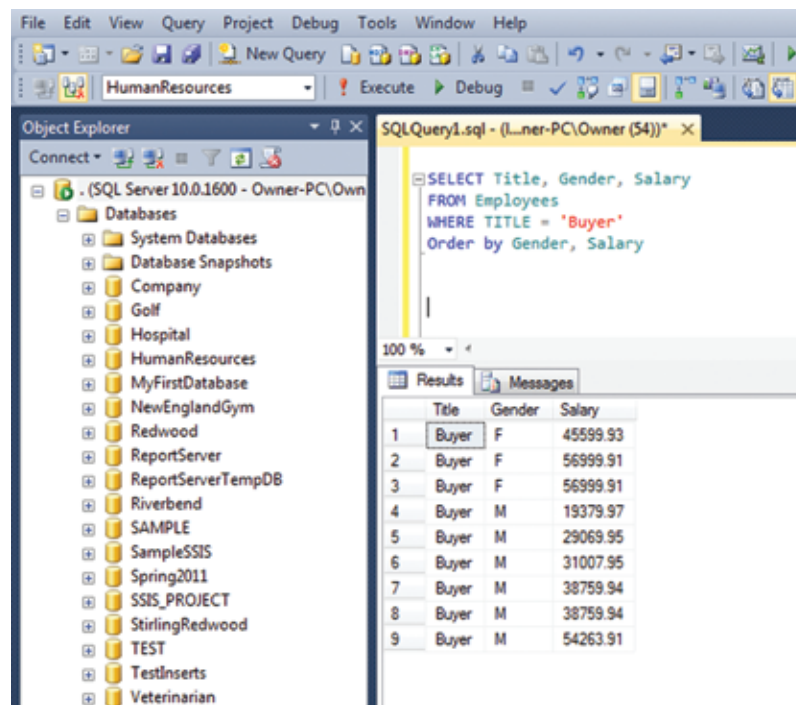


Figure 11

Notice that the GROUP BY clause contains the “Title” column. This is because we want the “Title” column to display in the result set so we are required to have it in the group by clause or an aggregate function error would occur as explained earlier.

Using the DISTINCT Keyword

The DISTINCT keyword can be used with aggregate functions so that values that are identical will be considered **only once** in the calculation. Before we see an example of using the DISTINCT keyword let us first look at the salaries for buyers. Figure 12 shows rows for employees who are buyers ordered by gender and salary.



The screenshot shows the SQL Server Enterprise Manager interface. The Object Explorer on the left displays a list of databases, including HumanResources. The central pane shows a query window with the following SQL code:

```
SELECT Title, Gender, Salary
FROM Employees
WHERE TITLE = 'Buyer'
Order by Gender, Salary
```

The Results pane at the bottom displays the query output as a table with 9 rows. The columns are Title, Gender, and Salary. The data is ordered by Gender and then Salary.

	Title	Gender	Salary
1	Buyer	F	45599.93
2	Buyer	F	56999.91
3	Buyer	F	56999.91
4	Buyer	M	19379.97
5	Buyer	M	29069.95
6	Buyer	M	31007.95
7	Buyer	M	38759.94
8	Buyer	M	38759.94
9	Buyer	M	54263.91

Figure 12

Notice that two female employees have identical salaries (56999.91) and two males have identical salaries (38759.94). Now if we use the same SELECT statement that was used earlier in Figure 11, but add the DISTINCT keyword the results would be different as shown in Figure 13.

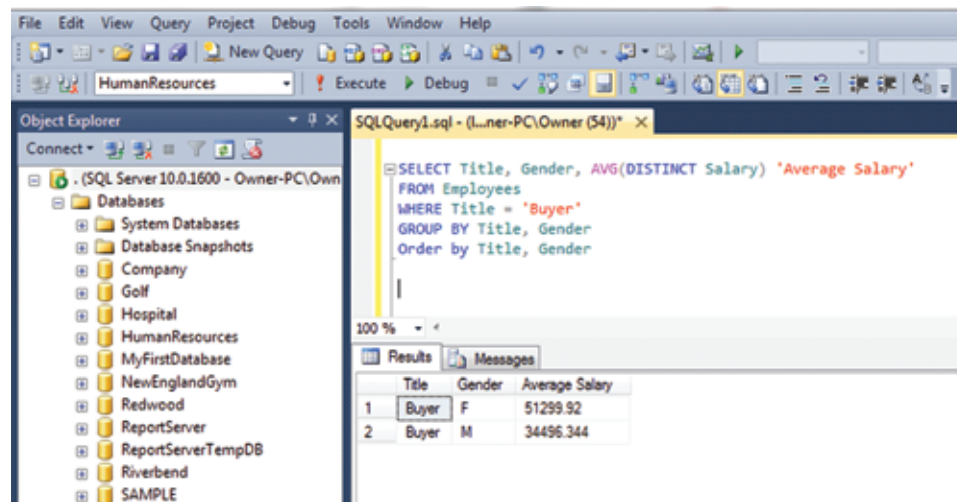


Figure 13

As you can see the results are different. That's because the values that are identical were only counted once in the example shown in Figure 13, whereas the example in Figure 12 shows all were counted, even the identical ones.

SUM() Function

The SUM() function sums up the values of a column. For example, if the manager wants to see the total salaries paid for each department the SELECT statement in Figure 14 could be executed.

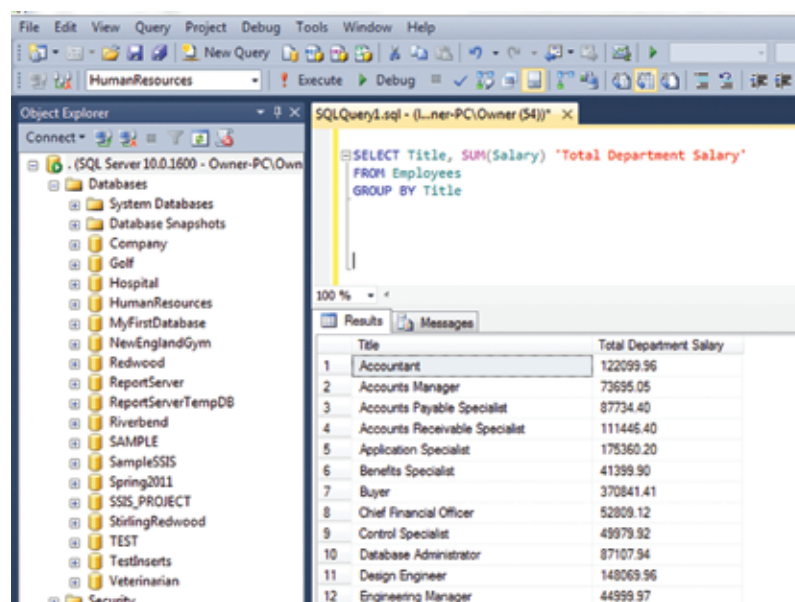


Figure 14

MIN() Function

The MIN() function returns the minimum value for a column. For example, if a manager wanted to know the minimum salary for each department the SELECT statement in Figure 15 could be executed.

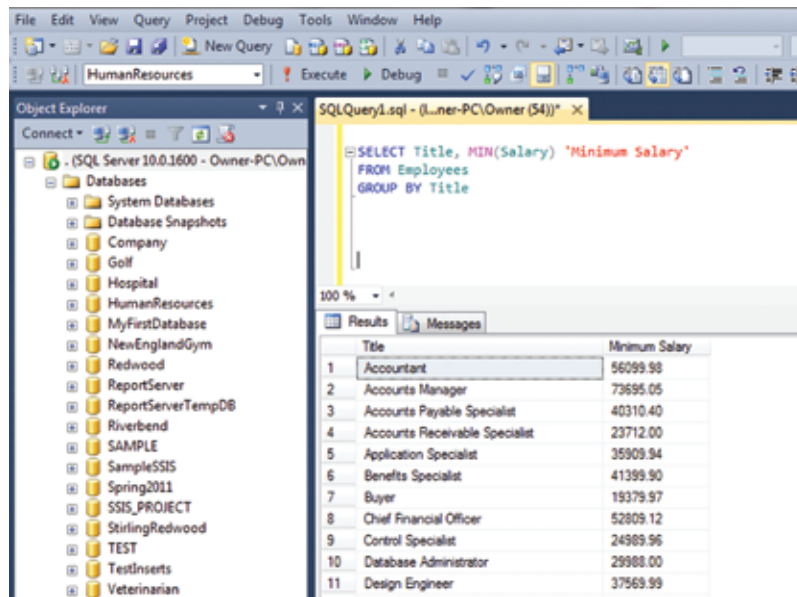


Figure 15

MAX() Function

The MAX() function is used to return the maximum value of a column. For example, if the manager wanted to see a list of the maximum salaries for each department the SELECT statement in Figure 16 could be used.

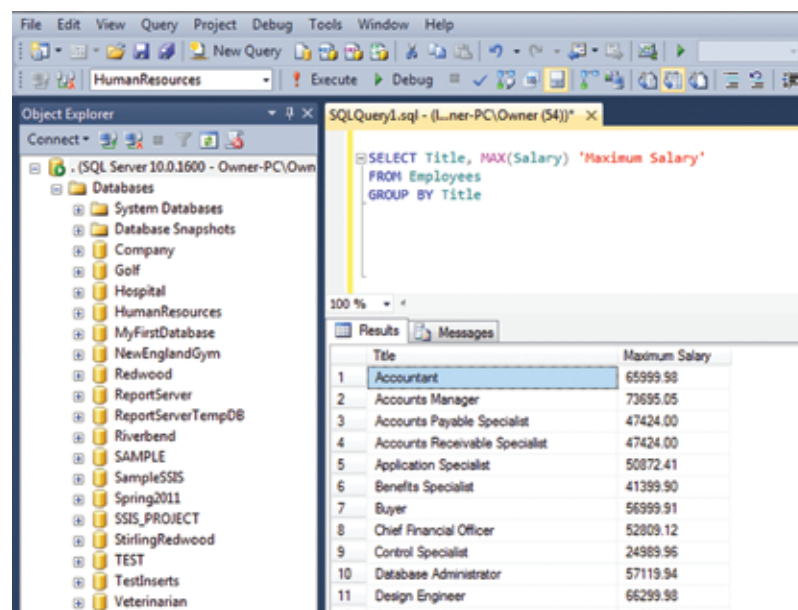


Figure 16

Summary

You have learned how to write SELECT statements to provide result sets that can help businesses make strategic decisions using aggregate functions. Various types of aggregate functions are available to produce informative results from the data to assist decision making. In addition, the HAVING clause allows filtering of aggregate data to drill down to specific summaries.

Exercises

1. Find the total number of single employees and the total number of married employees.
2. Find the titles and the number of employees with those titles, where the count of employees is greater than five.
3. Find the titles and the number of employees with those titles, where salaries are greater than or equal to \$50,000 and the total count of employees in those positions is greater than one.
4. List the titles where the average salaries are greater than \$75,000.
5. Show the sums of vacation hours and sick hours individually, grouped by title.
6. Provide a list of titles where the minimum amount of vacation hours is less than 10.
7. Provide a list of titles where the maximum number of sick hours is greater than 65.
8. Use the Departments table to find the total number of departments assigned to groups.
9. Use the Addresses table to find the number of employees that live in each state.
10. List counts for titles grouped and ordered by title, gender, and marital status for counts greater than three.
11. Retrieve the maximum salary for an accountant.
12. List the number of employees in each title where the total number of employees is less than three and the minimum salary for that title is greater than \$70,000.
13. Find the average salary for female workers in each title.
14. Show the sum of all salaries for the accountant title.
15. Find the three lowest paid Marketing Specialists.

