

The Girl Sprout® cookie sales leader problem

The troop leader of the local Girl Sprout troop is trying to figure out which girl sold the most cookies. So far she's got a table of each girl's sales for each day.

I need to find the winner soon. No one likes an angry Girl Sprout.



Edwina, confused Girl Sprout troop leader

Girl Sprout who made the sale → **cookie_sales** ← Dollar amount earned
Sales for this date

ID	first_name	sales	sale_date
1	Lindsay	32.02	3-6-2007
2	Paris	26.53	3-6-2007
3	Britney	11.25	3-6-2007
4	Nicole	18.96	3-6-2007
5	Lindsay	9.16	3-7-2007
6	Paris	1.52	3-7-2007
7	Britney	43.21	3-7-2007
8	Nicole	8.05	3-7-2007
9	Lindsay	17.62	3-8-2007
10	Paris	24.19	3-8-2007
11	Britney	3.40	3-8-2007
12	Nicole	15.21	3-8-2007
13	Lindsay	0	3-9-2007
14	Paris	31.99	3-9-2007
15	Britney	2.58	3-9-2007
16	Nicole	0	3-9-2007
17	Lindsay	2.34	3-10-2007
18	Paris	13.44	3-10-2007
19	Britney	8.78	3-10-2007
20	Nicole	26.82	3-10-2007
21	Lindsay	3.71	3-11-2007
22	Paris	.56	3-11-2007
23	Britney	34.19	3-11-2007
24	Nicole	7.77	3-11-2007
25	Lindsay	16.23	3-12-2007
26	Paris	0	3-12-2007
27	Britney	4.50	3-12-2007
28	Nicole	19.22	3-12-2007



Sharpen your pencil

The Girl Sprout with the largest total amount sold will win free horseback riding lessons. All of the Girl Sprouts want to win, so it's crucial that Edwina figure out the correct winner before things get ugly.

Use your new ORDER BY skills to write a query that will help Edwina find the name of the winner.

Sharpen your pencil Solution



The Girl Sprout with the largest total amount sold will win free horseback riding lessons. All of the Girl Sprouts want to win, so it's crucial that Edwina figure out the correct winner before things get ugly.

Use your new ORDER BY skills to write a query that will help Edwina find the name of the winner.

```
SELECT first_name, sales
FROM cookie_sales
ORDER BY first_name;
```

ORDER BY
Alphabetically orders your results based on a column you specify.

Here's our query...
...and here are the results.

first_name	sales
Britney	3.40
Britney	2.58
Britney	4.50
Britney	11.25
Britney	8.78
Britney	43.21
Britney	34.19
Lindsay	17.62
Lindsay	9.16
Lindsay	0.00
Lindsay	32.02
Lindsay	2.34
Lindsay	3.71
Lindsay	16.23
Nicole	19.22
Nicole	0.00
Nicole	8.05
Nicole	26.82
Nicole	7.77
Nicole	15.21
Nicole	18.96
Paris	26.53
Paris	0.00
Paris	0.56
Paris	1.52
Paris	13.44
Paris	24.19
Paris	31.99

107.91

81.08

96.03

98.23

The sales for each girl still had to be added together manually to find the winner..

SUM can add them for us

The stakes are high. We can't make a mistake and risk making our Girl Sprouts angry. Instead of adding these up ourselves, we can make SQL do the heavy lifting for us.

The SQL language has some special keywords, called *functions*. Functions are bits of code that perform an operation on a value or values. The first one we'll show you performs a mathematical operation on a column. We'll use the SUM function which works by **totaling the values in a column** designated by parentheses. Let's see it in action.

The SUM function totals the values in the sales column.

SUM is known as a function. This means that it performs an action on the column next to it that's in parentheses.

```
SELECT SUM(sales)
FROM cookie_sales
WHERE first_name = 'Nicole';
```

This restricts the query to only add up Nicole's sales. Otherwise it would be totaling the whole of the sales column.

```
File Edit Window Help TheWinnerIs
> SELECT SUM(sales) FROM cookie_sales
-> WHERE first_name = 'Nicole';
+-----+
| SUM(sales) |
+-----+
|      96.03 |
+-----+
1 row in set (0.00 sec)
```

The SUM() function returns the total sum of a numeric column. In the case of SUM, you must specify the field for which you want a total and the field must be numeric.

**Now we need the other three totals and we're done.
But it would be easier if we could do it in one single query...**



Exercise

Try this at home

Try it yourself. Create a table like the cookie_sales table and insert some decimal values in it. Then work through the queries you'll find over the next few pages.

SUM all of them at once with GROUP BY

There is a way to SUM each of the girl's sales at the same time. We'll just add a GROUP BY to our SUM statement. This groups all of the first name values for each girl and totals the sales for this group.

Grouping means creating groups of records that share some common characteristic.

```
SELECT first_name, SUM(sales)
FROM cookie_sales
GROUP BY first_name
ORDER BY SUM(sales) DESC;
```

SUM all the sales column figures.

Group together all the first_name values.

We have to order by the same SUM that we selected with.

We want the values displayed high-to-low so we can see the winner more easily.

This statement totals all the sales values in each first name group.

first_name	sales
Nicole	19.22
Nicole	0.00
Nicole	8.05
Nicole	26.82
Nicole	7.77
Nicole	15.21
Nicole	18.96

first_name	sales
Paris	26.53
Paris	0.00
Paris	0.56
Paris	1.52
Paris	13.44
Paris	24.19
Paris	31.99

first_name	sales
Lindsay	17.62
Lindsay	9.16
Lindsay	0.00
Lindsay	32.02
Lindsay	2.34
Lindsay	3.71
Lindsay	16.23

first_name	sales
Britney	3.40
Britney	2.58
Britney	4.50
Britney	11.25
Britney	8.78
Britney	43.21
Britney	34.19

And the winner is Britney! →

```
File Edit Window Help TheWinnerReallyIs
> SELECT first_name, SUM(sales)
-> FROM cookie_sales GROUP BY first_name
-> ORDER BY SUM(sales);
+-----+-----+
| first_name | sum(sales) |
+-----+-----+
| Britney    |      107.91 |
| Paris      |      98.23  |
| Nicole     |      96.03  |
| Lindsay    |      81.08  |
+-----+-----+
4 rows in set (0.00 sec)
```

AVG with GROUP BY

The other girls were disappointed, so Edwina has decided to give another prize to the girl with the highest daily average. She uses the AVG function.

Each girl has seven days of sales. For each girl, the AVG function adds together her sales and then divides it by 7.

WHOA! Hold up here. Write and SQL query to change the name of Nicole to Paris.

Then change the name of Ashley to Nicole

Again, we're grouping together all the first_name values...

... but this time we're averaging the value.

```
SELECT first_name, AVG(sales)
FROM cookie_sales
GROUP BY first_name;
```

AVG adds all of the values in a group and then divides by the total number of values to find the average value for that group.

first_name	sales
Nicole	19.22
Nicole	0.00
Nicole	8.05
Nicole	26.82
Nicole	7.77
Nicole	15.21
Nicole	18.96

first_name	sales
Paris	26.53
Paris	0.00
Paris	0.56
Paris	1.52
Paris	13.44
Paris	24.19
Paris	31.99

first_name	sales
Lindsay	17.62
Lindsay	9.16
Lindsay	0.00
Lindsay	32.02
Lindsay	2.34
Lindsay	3.71
Lindsay	16.23

first_name	sales
Britney	3.40
Britney	2.58
Britney	4.50
Britney	11.25
Britney	8.78
Britney	43.21
Britney	34.19

When rows are grouped, one line of output is produced for each group. Only statistics calculated for the group or fields whose values are the same for all rows in a group can be displayed in the grouped results.

```
File Edit Window Help TheWinnerReallyIs
> SELECT first_name, AVG(sales)
-> FROM cookie_sales GROUP BY first_name;
+-----+-----+
| first_name | AVG(sales) |
+-----+-----+
| Nicole     | 13.718571 |
| Britney    | 15.415714 |
| Lindsay    | 11.582857 |
| Paris      | 14.032857 |
+-----+-----+
4 rows in set (0.00 sec)
```

MIN and MAX

Not willing to leave anything out, Edwina takes a quick look at the MIN and MAX values from her table to see if any of the other girls had a larger sale value for a single day, or even if Britney had a worse day and got a lower value than any of the others...

We can use the function MAX to find the **largest value in a column**. MIN will give us the **smallest value in a column**.

```
SELECT first_name, MAX(sales)
FROM cookie_sales
GROUP BY first_name;
```

MAX returns the single largest sale value for each girl.

Surprise, Britney had the highest single day sales.

first_name	sales
Nicole	26.82
Britney	43.21
Lindsay	32.02
Paris	31.99

```
SELECT first_name, MIN(sales)
FROM cookie_sales
GROUP BY first_name;
```

MIN returns the single lowest sale value for each girl.

And while it looks like the other girls slacked off at least one day each, even on Britney's worst day she made money.

first_name	sales
Nicole	0.00
Britney	2.58
Lindsay	0.00
Paris	0.00

This is getting serious. Maybe I can give the prize to the girl who sold cookies on more days than any of the others.



COUNT the days

To figure out which girl sold cookies on more days than any other, Edwina tries to work out how many days the cookies were sold with the COUNT function. COUNT will return the **number of rows in a column.**

COUNT
Can tell you how many rows match a SELECT query without you having to see the rows. COUNT returns a single integer value.

```
SELECT COUNT(sale_date)
FROM cookie_sales;
```

COUNT returns the number of rows in the sale_date column. If the value is NULL, it isn't counted.

Sharpen your pencil

cookie_sales

ID	first_name	sales	sale_date
1	Lindsay	32.02	3-6-2007
2	Paris	26.53	3-6-2007
3	Britney	11.25	3-6-2007
4	Nicole	18.96	3-6-2007
5	Lindsay	9.16	3-7-2007
6	Paris	1.52	3-7-2007
7	Britney	43.21	3-7-2007
8	Nicole	8.05	3-7-2007
9	Lindsay	17.62	3-8-2007
10	Paris	24.19	3-8-2007
11	Britney	3.40	3-8-2007
12	Nicole	15.21	3-8-2007
13	Lindsay	0	3-9-2007
14	Paris	31.99	3-9-2007
15	Britney	2.58	3-9-2007
16	Nicole	0	3-9-2007
17	Lindsay	2.34	3-10-2007
18	Paris	13.44	3-10-2007
19	Britney	8.78	3-10-2007
20	Nicole	26.82	3-10-2007
21	Lindsay	3.71	3-11-2007
22	Paris	.56	3-11-2007
23	Britney	34.19	3-11-2007
24	Nicole	7.77	3-11-2007
25	Lindsay	16.23	3-12-2007
26	Paris	0	3-12-2007
27	Britney	4.50	3-12-2007
28	Nicole	19.22	3-12-2007

Here's the original table. What do you think will be returned by the query?

.....

.....

Does this number represent the actual number of days cookies were sold?

.....

.....

Write a query that will give us the number of days that each girl sold cookies.

```
SELECT first_name, count( sale_date )
FROM cookie_sales
WHERE sales > 0
GROUP BY first_name;
```



Sharpen your pencil Solution

Here's the original table. What do you think will be returned by the query?

28 sales dates

Does this number represent the actual number of days cookies were sold?

No. This number simply represents the number of values in the table for `sale_date`.

Write a query that will give us the number of days that each girl sold cookies.

```
SELECT first_name, COUNT(sale_date)
FROM cookie_sales
WHERE sales > 0
GROUP BY first_name;
```

This weeds out days that the girls sold no cookies.



You could just do an ORDER BY on the `sale_date` and look at the first and last dates to figure out how many days cookies were sold. Right?

Well, no. You couldn't be sure that there weren't days missing between the first and last dates.

There's a much easier way to find out the actual days that cookies were sold, and that's using the keyword `DISTINCT`. Not only can you use it to give you that `COUNT` you've been needing, but you can also get a list of the dates with no duplicates.

```
SELECT columnlist
FROM tablelist
WHERE condition
GROUP BY columnlist
HAVING condition
ORDER BY columnlist
```

It should be noted that, when using any of the above keywords in a `SELECT` statement, they need to be entered in the order shown. For example, `HAVING` keyword needs to always be after a `GROUP BY` but before an `ORDER BY`. page 107, The Language of SQL

DISTINCT

Returns each unique value only once, with no duplicates.

Q: What is the difference between the `ORDER BY` clause and the `GROUP BY` clause?

A: The `GROUP BY` clause is used to sort groups of data, and the `ORDER BY` clause is used to sort individual rows.

SELECT DISTINCT values

First let's look at that keyword `DISTINCT` *without* the `COUNT` function.

```
SELECT DISTINCT sale_date
FROM cookie_sales
ORDER BY sale_date;
```

Since `DISTINCT` is a keyword and not a function, you don't need parentheses around `sale_date`.

Here's our `ORDER BY` so we can see the first and last sales dates.

Look at that, not a duplicate in the bunch!

```
File Edit Window Help NoDups
> SELECT DISTINCT sale_date
FROM cookie_sales
-> ORDER BY sale_date;
+-----+
| sale_date |
+-----+
| 2007-03-06 |
| 2007-03-07 |
| 2007-03-08 |
| 2007-03-09 |
| 2007-03-10 |
| 2007-03-11 |
| 2007-03-12 |
+-----+
7 rows in set (0.00 sec)
```

Now let's try it *with* the `COUNT` function:

```
SELECT COUNT(DISTINCT sale_date)
FROM cookie_sales;
```

Notice that the `DISTINCT` goes inside the parentheses with `sale_date`.

We don't need an `ORDER BY` because `COUNT` will be returning a single number. Nothing to `ORDER` here.



BRAIN BARBELL

Try out this query, and then use it to figure out which girl sold cookies on the most days?

Answer: Britney

A bunch of SQL functions and keywords, in full costume, are playing a party game, “Who am I?” They’ll give you a clue—you try to guess who they are based on what they say. Assume they always tell the truth about themselves. Fill in the blanks to the right to identify the attendees. Also, for each attendee, write down whether it’s a function or keyword.

Tonight’s attendees:

COUNT, DISTINCT, AVG, MIN, GROUP BY, SUM, MAX



	Name	function or keyword
The result you get from using me might not be worth much.
What I spit out is larger than anything I take in.
I'll give you one-of-a-kind results.
I'll tell you how many there were.
You need to use me if you want to get a sum.
I'm only interested in the big number.
How am I? Somewhere in the middle.

Answers on page 279.

there are no Dumb Questions

Q: Since you were looking for the highest values with AVG, MAX, and MIN, couldn't you have added an ORDER BY clause?

A: We could have, and it would have been a very good idea. We chose to leave it out so as to not clutter up the queries and make it easier for you to learn the new functions. Take a look back over those functions and visualize the ORDER BY there. See how it would change the results?

Q: That DISTINCT keyword seems pretty useful. Can I use it with any column I want?

A: You can. It's especially useful when you have multiple records with the same value in a single column, and you simply want to see the variety of the values, and not a long list of duplicate values.

Q: Doing the query for MIN() didn't really have anything to do with Edwina finding a winner, did it?

A: No, but it would have helped her find the girls who did the worst. Next year, she can keep an eye on them to motivate them more.

Q: Speaking of MIN, what happens if there's a NULL in the column?

A: Good question. No, NULL is never returned by any of these functions, because NULL is the absence of a value, not the same thing as zero.



Hmm. AVG, MAX, and COUNT really didn't work out as a way to determine the second place winner. I wonder if I can use SUM to work out which girl came in second place and give her a prize.



Imagine we had not four, but **forty** Girl Sprouts. How could we use SUM to work out the second position?

LIMIT the number of results

Now we're going to use SUM to determine second place. Let's look back at the original query and results to help us figure out how to get that winner.

```
SELECT first_name, SUM(sales)
FROM cookie_sales
GROUP BY first_name
ORDER BY SUM(sales) DESC;
```

It's crucial that we use ORDER BY here; otherwise our results would be arbitrary.

first_name	sales
Britney	107.91
Paris	98.23
Nicole	96.03
Lindsay	81.08

We really only want the first two results.

Paris is our second-place winner! Nicole has stopped speaking to her.

Since we only have four results, it's easy to see who came in second place. But if we wanted to be even more precise, we could LIMIT the number of results just to the top two girls. That way we could see precisely the results we want. LIMIT allows us to specify exactly how many rows we want returned from our result set.

```
SELECT first_name, SUM(sales)
FROM cookie_sales
GROUP BY first_name
ORDER BY SUM(sales) DESC
LIMIT 2;
```

This is saying that you want to LIMIT your results to the first two.

It's a long query and gets you these two little results.

first_name	sales
Britney	107.91
Paris	98.23

While there are only four Girl Sprouts in the table and limiting it to two doesn't help a huge amount here, imagine that you were working with a much larger table. Suppose you had a list of the top 1,000 current songs playing at radio stations, but you wanted the top 100 in order of popularity. LIMIT would allow you to see only those and not the other 900 songs.

LIMIT to just second place

LIMIT even allows us to pinpoint the second place winner without having to see the first place winner. For this, we can use LIMIT with two parameters:

If you tried to guess what this would result in, you'd probably be wrong. When you have two parameters it means something completely different than with one.

LIMIT 0, 4

This is the result to start with. SQL starts counting with 0.

This is how many results to return.

first_name	sales
Britney	107.91
Paris	98.23
Nicole	96.03
Lindsay	81.08

Britney is 0, Paris is 1, Nicole is 2, and Lindsay is 3

Remember our top 100 songs? Suppose we wanted to see songs 20 through 29. Adding an extra parameter to our LIMIT would really help us. We'd simply be able to order them by popularity and add LIMIT 19, 10. The 19 says to start with the 20th song since SQL counts starting with 0, and the 10 says to give us back 10 rows.



Sharpen your pencil

Write the query that will get us the second result **and only the second result** using the LIMIT clause with two parameters.



Sharpen your pencil Solution

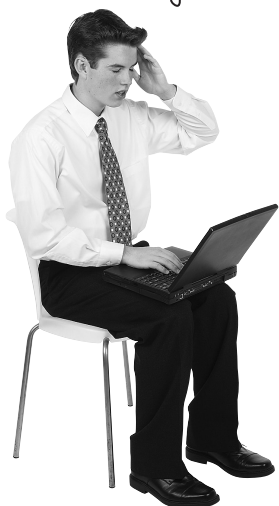
Write the query that will get us the second result **and only the second result** using the LIMIT clause with two parameters.

```
SELECT first_name, SUM(sales)
FROM cookie_sales
GROUP BY first_name
ORDER BY SUM(sales) DESC
LIMIT 1,1;
```

Remember, SQL starts counting with 0. So 1 is actually 2.

We don't count from zero, we index from zero. The first item, has an index of zero. So item one in a column is referred to as item[0].

My SQL statements are getting so long and complicated now, with all those new keywords. I like them, they're great, but isn't there a way I can simplify things?



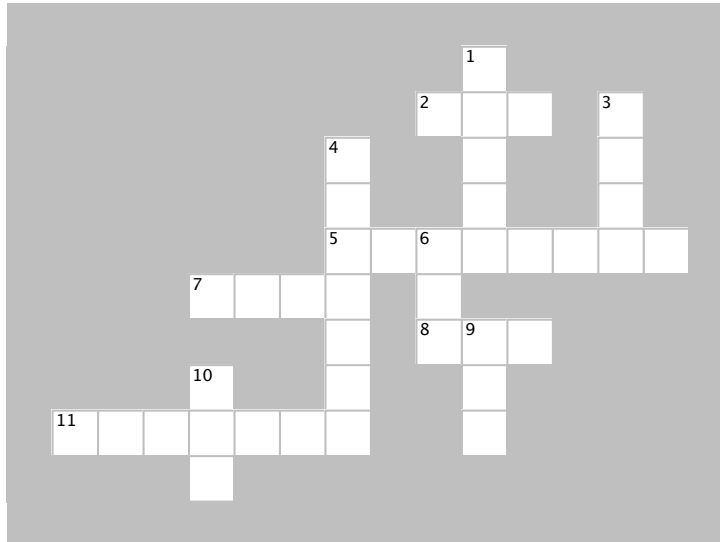
Your queries are getting longer because your data is getting more complicated.

Let's take a closer look at your table, you may have outgrown it. Move along to Chapter 7...



SELECTcross

It's time to give your right brain a break and put that left brain to work: all the words are SQL-related and from this chapter.

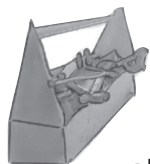


Across

2. You can find the smallest value in a column with this function.
5. This keyword returns each unique value only once, with no duplicates.
7. The _____ keyword in the CASE allows you to tell your RDBMS what to do if any records don't meet the conditions
8. You can find the largest value in a column with this function.
11. Use these two words to consolidate rows based on a common column.

Down

1. Lets you specify exactly how many rows to return, and which row to start with.
3. If you ORDER BY a column using this keyword, the value 9 in that column will come before 8.
4. Use these two words to alphabetically order your results based on a column you specify.
6. This function adds up a column of numeric values.
9. If you ORDER BY a column using this keyword, the value 8 in that column will come before 9.
10. Use this in a SELECT to return the total value of results rather than the results themselves.



Your SQL Toolbox

You've got Chapter 6 under your belt, and you're really cruising now with all those advanced **SELECT** functions, keywords, and queries. For a complete list of tooltips in the book, see Appendix iii.

ORDER BY

Alphabetically orders your results based on a column you specify.

GROUP BY

Consolidates rows based on a common column.

COUNT

Can tell you how many rows match a **SELECT** query without you having to see the rows. **COUNT** returns a single integer value.

DISTINCT

Returns each unique value only once, with no duplicates

SUM

Adds up a column of numeric values.

AVG

Returns the average value in a numeric column.

MAX and MIN

Return the largest value in a column with **MAX**, and the smallest with **MIN**.

LIMIT

Lets you specify exactly how many rows to return, and which row to start with.

Your new tools:
advanced **SELECT**
functions, keywords,
and queries!