7.3.4 Use Count, Group By, and Order By

There is a lot that goes into using joins, but practice certainly helps. Learning aliases has made our code cleaner, too. Thanks to the extra lists we've been working on, Bobby understands more about how joins work in SQL. He can put that knowledge into action by joining the correct tables to create new lists.

But before he's able to hand that list over, he'll need to use `COUNT` and `GROUP BY` with joins to separate the employees into their departments. These two components are very similar to functions used in Pandas. `COUNT` will count the rows of data in a dataset, and we can use `GROUP BY` to group our data by type. If the boss were to ask Bobby how many employees are retiring from the Sales department, we would use both of these functions together with joins to generate an answer.

We can also arrange the data so it presents itself in descending or ascending order by using another function: `ORDER BY`. This is another function that appears in Pandas and works the same way.

Let's continue to work on these lists with Bobby, this time using `COUNT`, `GROUP BY`, and `ORDER BY` with the joins.

We know that we'll need to use some joins to organize the data we need in one table. We also know that Bobby needs a count of employees for each department.

To organize the counts, we'll need to add a `GROUP BY` clause to our select statement. In Postgres, `GROUP BY` is used when we want to group rows of identical data together in a table. It is precisely the clause we want to group separate employees into their departments.

In your query editor, join the current_emp and dept_emp tables. The new code should be added at the bottom, below the existing code.

```
-- Employee count by department number
SELECT COUNT(ce.emp_no), de.dept_no
FROM current_emp as ce
LEFT JOIN dept_emp as de
ON ce.emp_no = de.emp_no
GROUP BY de.dept_no;
```

A few things to note:

- The `COUNT` function was used on the employee numbers.
- Aliases were assigned to both tables.
- `GROUP BY` was added to the `SELECT` statement.

We added `COUNT()` to the `SELECT` statement because we wanted a total number of employees. We couldn't actually use the `SUM()` function because the employee numbers would simply be added, which would leave us with one really large and useless number.

Bobby's boss asked for a list of how many employees per department were leaving, so the only columns we really needed for this list were the employee number and the department number.

We used a `LEFT JOIN` in this query because we wanted all employee numbers from Table 1 to be included in the returned data. Also, if any employee numbers weren't assigned a department number, that would be made apparent.

The `ON` portion of the query tells Postgres which columns we're using to match the data. Both tables have an emp_no column, so we're using that to match the records from both tables.

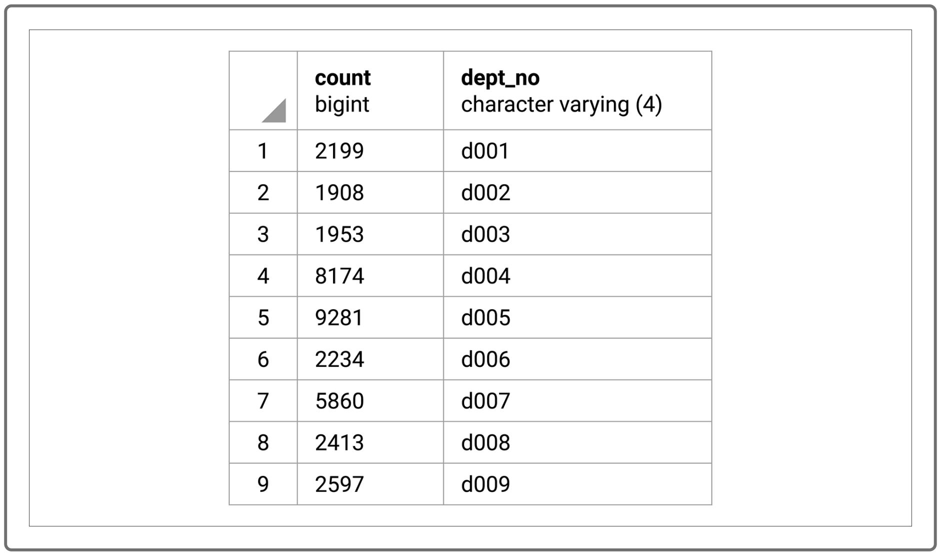
`GROUP BY` is the magic clause that gives us the number of employees retiring from each department.

Did you notice that the data output isn't in any particular order? In fact, if you executed the code again, the same numbers would be returned in another order altogether. Thankfully, there is one additional clause we can add to the query to keep everything in order: `ORDER BY`.

`ORDER BY` does exactly as it reads: It puts the data output in order for us. Let's update our code. In the query editor, add the `ORDER BY` line to the end of the code block, so your code looks like the following:

```sql
-- Employee count by department number
SELECT COUNT(ce.emp_no), de.dept_no
FROM current_emp as ce
LEFT JOIN dept_emp as de
ON ce.emp_no = de.emp_no
GROUP BY de.dept_no
ORDER BY de.dept_no;
```

Now, instead of a randomly ordered output each time, it will be organized by the department number.



	count bigint	dept_no character varying (4)
1	2199	d001
2	1908	d002
3	1953	d003
4	8174	d004
5	9281	d005
6	2234	d006
7	5860	d007
8	2413	d008
9	2597	d009

Now test your skills with the following Skill Drill.

SKILL DRILL

Bobby can present this table to his boss since it provides a breakdown for each department. Update the code block to create a new table, then export it as a CSV.