```
De Haan          Executive              5424
Kochhar          Executive              6634
Chen             Finance                3705
Faviet           Finance                4844
Popp             Finance                2905
Sciarra          Finance                3703
Urman            Finance                3545
Austin           IT                     3800
9 rows selected.
```

The subquery can of course include a WHERE clause to restrict the rows inserted into the new table. To create a table with no rows, use a WHERE clause that will exclude all rows:

```
create table no_emps as select * from employees where 1=2;
```

The WHERE clause 1=2 can never return TRUE, so the table structure will be created ready for use, but no rows will be inserted at creation time.

Altering Table Definitions after Creation

There are many alterations that can be made to a table after creation. Those that affect the physical storage fall into the domain of the database administrator, but many changes are purely logical and will be carried out by the SQL developers. The following are examples (for the most part self-explanatory):

- Adding columns:

```
alter table emp add (job_id number);
```

- Modifying columns:

```
alter table emp modify (comm number(4,2) default 0.05);
```

- Dropping columns:

```
alter table emp drop column comm;
```

- Marking columns as unused:

```
alter table emp set unused column job_id;
```

- Renaming columns:

```
alter table emp rename column hiredate to recruited;
```

- Marking the table as read-only:

```
alter table emp read only;
```

All of these changes are DDL commands with the built-in COMMIT. They are therefore nonreversible and will fail if there is an active transaction against the table. They are also virtually instantaneous with the exception of dropping a column. Dropping a column can be a time-consuming exercise because as each column is dropped, every row must be restructured to remove the column's data. The SET UNUSED command, which makes columns nonexistent as far as SQL is concerned, is often a better alternative, followed when convenient by

```
ALTER TABLE tablename DROP UNUSED COLUMNS;
```

which will drop all the unused columns in one pass through the table.

Marking a table as read-only will cause errors for any attempted DML commands. But the table can still be dropped. This can be disconcerting but is perfectly logical when you think it through. A DROP command doesn't actually affect the table: it affects the tables in the data dictionary that define the table, and these are not read-only.

Dropping and Truncating Tables

The TRUNCATE TABLE command was described in Chapter 10: it has the effect of removing every row from a table, while leaving the table definition intact. DROP TABLE is more drastic in that the table definition is removed as well. The syntax is as follows:

DROP TABLE [*schema.*]*tablename* ;

If *schema* is not specified, then the table called *tablename* in your currently logged on schema will be dropped.

As with a TRUNCATE, SQL will not produce a warning before the table is dropped, and furthermore, as with any DDL command, it includes a COMMIT. A DROP is therefore absolutely nonreversible. But there are some restrictions: if any session (even your own) has a transaction in progress that includes a row in the table, then the DROP will fail, and it is also impossible to drop a table that is referred to in a foreign key constraint defined for a another table. This table (or the constraint) must be dropped first.