Here I would suggest digging into these few lines of code, and evaluating the individual steps to make sure you understand exactly what they are doing to the result. It's certainly a somewhat complicated example, but understanding these pieces will give you the means to similarly explore your own data.

Pivot Tables

We have seen how the `GroupBy` abstraction lets us explore relationships within a dataset. A *pivot table* is a similar operation that is commonly seen in spreadsheets and other programs that operate on tabular data. The pivot table takes simple column-wise data as input, and groups the entries into a two-dimensional table that provides a multidimensional summarization of the data. The difference between pivot tables and `GroupBy` can sometimes cause confusion; it helps me to think of pivot tables as essentially a *multidimensional* version of `GroupBy` aggregation. That is, you split-apply-combine, but both the split and the combine happen across not a one-dimensional index, but across a two-dimensional grid.

Motivating Pivot Tables

For the examples in this section, we'll use the database of passengers on the *Titanic*, available through the Seaborn library (see "Visualization with Seaborn" on page 311):

```
In[1]: import numpy as np
       import pandas as pd
       import seaborn as sns
       titanic = sns.load_dataset('titanic')
```

```
In[2]: titanic.head()
```

```
Out[2]:
   survived  pclass     sex   age  sibsp  parch     fare embarked  class  \\
0         0       3    male  22.0      1      0   7.2500        S  Third
1         1       1  female  38.0      1      0  71.2833        C  First
2         1       3  female  26.0      0      0   7.9250        S  Third
3         1       1  female  35.0      1      0  53.1000        S  First
4         0       3    male  35.0      0      0   8.0500        S  Third

     who adult_male deck  embark_town alive  alone
0    man       True  NaN  Southampton    no  False
1  woman      False    C    Cherbourg   yes  False
2  woman      False  NaN  Southampton   yes   True
3  woman      False    C  Southampton   yes  False
4    man       True  NaN  Southampton    no   True
```

This contains a wealth of information on each passenger of that ill-fated voyage, including gender, age, class, fare paid, and much more.

Pivot Tables by Hand

To start learning more about this data, we might begin by grouping it according to gender, survival status, or some combination thereof. If you have read the previous section, you might be tempted to apply a GroupBy operation—for example, let's look at survival rate by gender:

```
In[3]: titanic.groupby('sex')[['survived']].mean()
```

```
Out[3]:         survived
        sex
        female  0.742038
        male    0.188908
```

This immediately gives us some insight: overall, three of every four females on board survived, while only one in five males survived!

This is useful, but we might like to go one step deeper and look at survival by both sex and, say, class. Using the vocabulary of GroupBy, we might proceed using something like this: we *group by* class and gender, *select* survival, *apply* a mean aggregate, *combine* the resulting groups, and then *unstack* the hierarchical index to reveal the hidden multidimensionality. In code:

```
In[4]: titanic.groupby(['sex', 'class'])['survived'].aggregate('mean').unstack()
```

```
Out[4]: class      First    Second     Third
        sex
        female  0.968085  0.921053  0.500000
        male    0.368852  0.157407  0.135447
```

This gives us a better idea of how both gender and class affected survival, but the code is starting to look a bit garbled. While each step of this pipeline makes sense in light of the tools we've previously discussed, the long string of code is not particularly easy to read or use. This two-dimensional GroupBy is common enough that Pandas includes a convenience routine, pivot_table, which succinctly handles this type of multidimensional aggregation.

Pivot Table Syntax

Here is the equivalent to the preceding operation using the pivot_table method of DataFrames:

```
In[5]: titanic.pivot_table('survived', index='sex', columns='class')
```

```
Out[5]: class      First    Second     Third
        sex
        female  0.968085  0.921053  0.500000
        male    0.368852  0.157407  0.135447
```

This is eminently more readable than the GroupBy approach, and produces the same result. As you might expect of an early 20th-century transatlantic cruise, the survival