```
df6                df7              pd.merge(df6, df7)
    name   food         name drink       name   food  drink
0  Peter   fish     0    Mary  wine   0   Mary  bread   wine
1   Paul  beans     1  Joseph  beer
2   Mary  bread
```

Here we have merged two datasets that have only a single "name" entry in common: Mary. By default, the result contains the *intersection* of the two sets of inputs; this is what is known as an *inner join*. We can specify this explicitly using the `how` keyword, which defaults to `'inner'`:

```
In[14]: pd.merge(df6, df7, how='inner')

Out[14]:    name   food drink
         0  Mary  bread  wine
```

Other options for the `how` keyword are `'outer'`, `'left'`, and `'right'`. An *outer join* returns a join over the union of the input columns, and fills in all missing values with NAs:

```
In[15]: print(df6); print(df7); print(pd.merge(df6, df7, how='outer'))

df6                df7              pd.merge(df6, df7, how='outer')
     name   food         name drink        name   food drink
0  Peter   fish     0    Mary  wine   0   Peter   fish   NaN
1   Paul  beans     1  Joseph  beer   1    Paul  beans   NaN
2   Mary  bread                      2    Mary  bread  wine
                                     3  Joseph    NaN  beer
```

The *left join* and *right join* return join over the left entries and right entries, respectively. For example:

```
In[16]: print(df6); print(df7); print(pd.merge(df6, df7, how='left'))

df6                df7              pd.merge(df6, df7, how='left')
     name   food         name drink        name   food drink
0  Peter   fish     0    Mary  wine   0   Peter   fish   NaN
1   Paul  beans     1  Joseph  beer   1    Paul  beans   NaN
2   Mary  bread                      2    Mary  bread  wine
```

The output rows now correspond to the entries in the left input. Using `how='right'` works in a similar manner.

All of these options can be applied straightforwardly to any of the preceding join types.

Overlapping Column Names: The suffixes Keyword

Finally, you may end up in a case where your two input `DataFrame`s have conflicting column names. Consider this example:

```
In[17]: df8 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'],
                            'rank': [1, 2, 3, 4]})
```

```
df9 = pd.DataFrame({'name': ['Bob', 'Jake', 'Lisa', 'Sue'],
                    'rank': [3, 1, 4, 2]})
print(df8); print(df9); print(pd.merge(df8, df9, on="name"))
```

```
df8                df9                pd.merge(df8, df9, on="name")
    name  rank         name  rank          name  rank_x  rank_y
0   Bob      1      0   Bob     3      0   Bob       1       3
1  Jake      2      1  Jake     1      1  Jake       2       1
2  Lisa      3      2  Lisa     4      2  Lisa       3       4
3   Sue      4      3   Sue     2      3   Sue       4       2
```

Because the output would have two conflicting column names, the merge function automatically appends a suffix `_x` or `_y` to make the output columns unique. If these defaults are inappropriate, it is possible to specify a custom suffix using the `suffixes` keyword:

```
In[18]:
print(df8); print(df9);
print(pd.merge(df8, df9, on="name", suffixes=["_L", "_R"]))
```

```
df8                df9
    name  rank         name  rank
0   Bob      1      0   Bob     3
1  Jake      2      1  Jake     1
2  Lisa      3      2  Lisa     4
3   Sue      4      3   Sue     2


pd.merge(df8, df9, on="name", suffixes=["_L", "_R"])
   name  rank_L  rank_R
0   Bob       1       3
1  Jake       2       1
2  Lisa       3       4
3   Sue       4       2
```

These suffixes work in any of the possible join patterns, and work also if there are multiple overlapping columns.

For more information on these patterns, see "Aggregation and Grouping" on page 158, where we dive a bit deeper into relational algebra. Also see the "Merge, Join, and Concatenate" section of the Pandas documentation for further discussion of these topics.

Example: US States Data

Merge and join operations come up most often when one is combining data from different sources. Here we will consider an example of some data about US states and their populations. The data files can be found at *http://github.com/jakevdp/data-USstates/*:

```
In[19]:
# Following are shell commands to download the data
```