In [ ]:
```
import pandas as pd
import numpy as np

class display(object):
    """Display HTML representation of multiple objects"""
    template = """<div style="float: left; padding: 10px;">
    <p style='font-family:"Courier New", Courier, monospace'>{0}</p>{1}
    </div>"""
    def __init__(self, *args):
        self.args = args

    def _repr_html_(self):
        return '\n'.join(self.template.format(a, eval(a)._repr_html_())
                         for a in self.args)

    def __repr__(self):
        return '\n\n'.join(a + '\n' + repr(eval(a))
                           for a in self.args)
```

Example: US States Data

Merge and join operations come up most often when 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:

Let's take a look at the three datasets, using the Pandas `read_csv` function:

In [ ]:
```
pop = pd.read_csv('state-population.csv')
areas = pd.read_csv('state-areas.csv')
abbrevs = pd.read_csv('state-abbrevs.csv')

display('pop.head()', 'areas.head()', 'abbrevs.head()')
```

Out[ ]:

`pop.head()`

	state/region	ages	year	population
0	AL	under18	2012	1117489.0
1	AL	total	2012	4817528.0
2	AL	under18	2010	1130966.0
3	AL	total	2010	4785570.0
4	AL	under18	2011	1125763.0

`areas.head()`

	state	area (sq. mi)
0	Alabama	52423
1	Alaska	656425
2	Arizona	114006
3	Arkansas	53182
4	California	163707

`abbrevs.head()`

	state	abbreviation
0	Alabama	AL
1	Alaska	AK
2	Arizona	AZ
3	Arkansas	AR
4	California	CA

Given this information, say we want to compute a relatively straightforward result: rank US states and territories by their 2010 population density. We clearly have the data here to find this result, but we'll have to combine the datasets to do so.

We'll start with a many-to-one merge that will give us the full state names within the population `DataFrame`. We want to merge based on the `state/region` column of `pop` and the `abbreviation` column of `abbrevs`. We'll use `how='outer'` to make sure no data is thrown away due to mismatched labels:

In [ ]:

```
merged = pop.merge(abbrevs,left_on='state/region',right_on='abbreviation',how='outer')
# merged = merged[merged['year'] == 2010]
merged = merged.drop_duplicates()                                    # drop column having duplicate info after merging
# merged = merged.sort_values(by='population',ascending=False)
merged.head()
```

Out[ ]:

	state/region	ages	year	population	state	abbreviation
0	AL	under18	2012	1117489.0	Alabama	AL
1	AL	total	2012	4817528.0	Alabama	AL
2	AL	under18	2010	1130966.0	Alabama	AL
3	AL	total	2010	4785570.0	Alabama	AL
4	AL	under18	2011	1125763.0	Alabama	AL

Let's double-check whether there were any mismatches here, which we can do by looking for rows with nulls:

In [ ]:
```
merged[merged.isna()]
```

Out[ ]:

	state/region	ages	year	population	state	abbreviation
0	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...
2539	NaN	NaN	NaN	NaN	NaN	NaN
2540	NaN	NaN	NaN	NaN	NaN	NaN
2541	NaN	NaN	NaN	NaN	NaN	NaN
2542	NaN	NaN	NaN	NaN	NaN	NaN
2543	NaN	NaN	NaN	NaN	NaN	NaN

2544 rows × 6 columns

Some of the `population` values are null; let's figure out which these are!

In [ ]:
```
merged[merged['population'].isna()]
```

Out[ ]:

	state/region	ages	year	population	state	abbreviation
2448	PR	under18	1990	NaN	NaN	NaN
2449	PR	total	1990	NaN	NaN	NaN
2450	PR	total	1991	NaN	NaN	NaN
2451	PR	under18	1991	NaN	NaN	NaN
2452	PR	total	1993	NaN	NaN	NaN
2453	PR	under18	1993	NaN	NaN	NaN
2454	PR	under18	1992	NaN	NaN	NaN
2455	PR	total	1992	NaN	NaN	NaN
2456	PR	under18	1994	NaN	NaN	NaN
2457	PR	total	1994	NaN	NaN	NaN
2458	PR	total	1995	NaN	NaN	NaN
2459	PR	under18	1995	NaN	NaN	NaN
2460	PR	under18	1996	NaN	NaN	NaN
2461	PR	total	1996	NaN	NaN	NaN
2462	PR	under18	1998	NaN	NaN	NaN
2463	PR	total	1998	NaN	NaN	NaN
2464	PR	total	1997	NaN	NaN	NaN
2465	PR	under18	1997	NaN	NaN	NaN
2466	PR	total	1999	NaN	NaN	NaN
2467	PR	under18	1999	NaN	NaN	NaN

It appears that all the null population values are from Puerto Rico prior to the year 2000; this is likely due to this data not being available in the original source.

More importantly, we see that some of the new `state` entries are also null, which means that there was no corresponding entry in the `abbrevs` key! Let's figure out which regions lack this match:

In [ ]:
```
merged[merged['state'].isna()]
```

Out[ ]:

	state/region	ages	year	population	state	abbreviation
2448	PR	under18	1990	NaN	NaN	NaN
2449	PR	total	1990	NaN	NaN	NaN
2450	PR	total	1991	NaN	NaN	NaN
2451	PR	under18	1991	NaN	NaN	NaN
2452	PR	total	1993	NaN	NaN	NaN
...	...	...	...	...	...	...
2539	USA	total	2010	309326295.0	NaN	NaN
2540	USA	under18	2011	73902222.0	NaN	NaN
2541	USA	total	2011	311582564.0	NaN	NaN
2542	USA	under18	2012	73708179.0	NaN	NaN
2543	USA	total	2012	313873685.0	NaN	NaN

96 rows × 6 columns

We can quickly infer the issue: our population data includes entries for Puerto Rico (PR) and the United States as a whole (USA), while these entries do not appear in the state abbreviation key. We can fix these quickly by filling in appropriate entries:

In [ ]:
```
pr = merged[merged['state/region'] == 'PR']
pr.loc[:,['state','abbreviation']] = ['Puerto Rico','PR']
```

```
merged[merged['state/region'] == 'PR'] = pr

usa = merged[merged['state/region'] == 'USA']
usa.loc[:,['state','abbreviation']] = ['United States of America','USA']
merged[merged['state/region'] == 'USA'] = usa

merged.tail()
```

Out[ ]:

	state/region	ages	year	population	state	abbreviation
2539	USA	total	2010	309326295.0	United States of America	USA
2540	USA	under18	2011	73902222.0	United States of America	USA
2541	USA	total	2011	311582564.0	United States of America	USA
2542	USA	under18	2012	73708179.0	United States of America	USA
2543	USA	total	2012	313873685.0	United States of America	USA

No more nulls in the `state` column: we're all set!

Now we can merge the result with the area data using a similar procedure. Examining our results, we will want to join on the `state` column in both:

In [ ]:
```
final = merged.merge(areas,left_on='state',right_on='state',how='outer')
final.head()
```

Out[ ]:

	state/region	ages	year	population	state	abbreviation	area (sq. mi)
0	AL	under18	2012	1117489.0	Alabama	AL	52423.0
1	AL	total	2012	4817528.0	Alabama	AL	52423.0
2	AL	under18	2010	1130966.0	Alabama	AL	52423.0
3	AL	total	2010	4785570.0	Alabama	AL	52423.0
4	AL	under18	2011	1125763.0	Alabama	AL	52423.0

Again, let's check for nulls to see if there were any mismatches:

In [ ]:
```
final.isna()
```

Out[ ]:

	state/region	ages	year	population	state	abbreviation	area (sq. mi)
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
2539	False	False	False	False	False	False	True
2540	False	False	False	False	False	False	True
2541	False	False	False	False	False	False	True
2542	False	False	False	False	False	False	True
2543	False	False	False	False	False	False	True

2544 rows × 7 columns

There are nulls in the `area` column; we can take a look to see which regions were ignored here:

In [ ]:
```
final[final['area (sq. mi)'].isna()]
```

Out[ ]:

	state/region	ages	year	population	state	abbreviation	area (sq. mi)
2496	USA	under18	1990	64218512.0	United States of America	USA	NaN
2497	USA	total	1990	249622814.0	United States of America	USA	NaN
2498	USA	total	1991	252980942.0	United States of America	USA	NaN
2499	USA	under18	1991	65313018.0	United States of America	USA	NaN
2500	USA	under18	1992	66509177.0	United States of America	USA	NaN
2501	USA	total	1992	256514231.0	United States of America	USA	NaN
2502	USA	total	1993	259918595.0	United States of America	USA	NaN
2503	USA	under18	1993	67594938.0	United States of America	USA	NaN
2504	USA	under18	1994	68640936.0	United States of America	USA	NaN
2505	USA	total	1994	263125826.0	United States of America	USA	NaN
2506	USA	under18	1995	69473140.0	United States of America	USA	NaN
2507	USA	under18	1996	70233512.0	United States of America	USA	NaN
2508	USA	total	1995	266278403.0	United States of America	USA	NaN
2509	USA	total	1996	269394291.0	United States of America	USA	NaN
2510	USA	total	1997	272646932.0	United States of America	USA	NaN
2511	USA	under18	1997	70920738.0	United States of America	USA	NaN
2512	USA	under18	1998	71431406.0	United States of America	USA	NaN
2513	USA	total	1998	275854116.0	United States of America	USA	NaN
2514	USA	under18	1999	71946051.0	United States of America	USA	NaN
2515	USA	total	2000	282162411.0	United States of America	USA	NaN
2516	USA	under18	2000	72376189.0	United States of America	USA	NaN
2517	USA	total	1999	279040181.0	United States of America	USA	NaN

	state/region	ages	year	population	state	abbreviation	area (sq. mi)
2518	USA	total	2001	284968955.0	United States of America	USA	NaN
2519	USA	under18	2001	72671175.0	United States of America	USA	NaN
2520	USA	total	2002	287625193.0	United States of America	USA	NaN
2521	USA	under18	2002	72936457.0	United States of America	USA	NaN
2522	USA	total	2003	290107933.0	United States of America	USA	NaN
2523	USA	under18	2003	73100758.0	United States of America	USA	NaN
2524	USA	total	2004	292805298.0	United States of America	USA	NaN
2525	USA	under18	2004	73297735.0	United States of America	USA	NaN
2526	USA	total	2005	295516599.0	United States of America	USA	NaN
2527	USA	under18	2005	73523669.0	United States of America	USA	NaN
2528	USA	total	2006	298379912.0	United States of America	USA	NaN
2529	USA	under18	2006	73757714.0	United States of America	USA	NaN
2530	USA	total	2007	301231207.0	United States of America	USA	NaN
2531	USA	under18	2007	74019405.0	United States of America	USA	NaN
2532	USA	total	2008	304093966.0	United States of America	USA	NaN
2533	USA	under18	2008	74104602.0	United States of America	USA	NaN
2534	USA	under18	2013	73585872.0	United States of America	USA	NaN
2535	USA	total	2013	316128839.0	United States of America	USA	NaN
2536	USA	total	2009	306771529.0	United States of America	USA	NaN
2537	USA	under18	2009	74134167.0	United States of America	USA	NaN
2538	USA	under18	2010	74119556.0	United States of America	USA	NaN
2539	USA	total	2010	309326295.0	United States of America	USA	NaN

	state/region	ages	year	population	state	abbreviation	area (sq. mi)
2540	USA	under18	2011	73902222.0	United States of America	USA	NaN
2541	USA	total	2011	311582564.0	United States of America	USA	NaN
2542	USA	under18	2012	73708179.0	United States of America	USA	NaN
2543	USA	total	2012	313873685.0	United States of America	USA	NaN

We see that our `areas` `DataFrame` does not contain the area of the United States as a whole. We could insert the appropriate value (using the sum of all state areas, for instance), but in this case we'll just drop the null values because the population density of the entire United States is not relevant to our current discussion:

In [ ]:
```
final.dropna(axis=0,inplace=True)
final[final['area (sq. mi)'].isna()]
```

Out[ ]:

state/region	ages	year	population	state	abbreviation	area (sq. mi)

Now we have all the data we need. To answer the question of interest, let's first select the portion of the data corresponding with the year 2010, and the total population. We'll use the `query` function to do this quickly (this requires the NumExpr package to be installed; see High-Performance Pandas: `eval()` and `query()`):

In [ ]:
```
import numexpr as ne
final = final[final['year'] == 2010]
final = final[final['ages'] == 'total']
```

Now let's compute the population density and display it in order. We'll start by re-indexing our data on the state, and then compute the result:

In [ ]:
```
final.set_index('state',inplace=True)
```

In [ ]:
```
final['density'] = final['population']/final['area (sq. mi)']
final.sort_values('density',ascending=False,inplace=True)
```

The result is a ranking of US states, plus Washington, DC, and Puerto Rico, in order of their 2010 population density, in residents per square mile. We can see that by far the densest region in this dataset is Washington, DC (i.e., the District of Columbia); among states, the densest is

New Jersey.

We can also check the end of the list:

In [ ]:
```
final.tail()
```

Out[ ]:

state	state/region	ages	year	population	abbreviation	area (sq. mi)	density
South Dakota	SD	total	2010	816211.0	SD	77121.0	10.583512
North Dakota	ND	total	2010	674344.0	ND	70704.0	9.537565
Montana	MT	total	2010	990527.0	MT	147046.0	6.736171
Wyoming	WY	total	2010	564222.0	WY	97818.0	5.768079
Alaska	AK	total	2010	713868.0	AK	656425.0	1.087509

We see that the least dense state, by far, is Alaska, averaging slightly over one resident per square mile.

This type of data merging is a common task when trying to answer questions using real-world data sources. I hope that this example has given you an idea of some of the ways you can combine the tools we've covered in order to gain insight from your data!