Part II: Joining

Topic – Joining

Sometimes data you want is separated over two tables. For example, if you worked for an airline, you might have a table that contains information about each passenger and a table about information for each flight. The flight information table likely contains a column that lists each passenger on the flight. The passenger information table likely lists the home address of each passenger. What if you wanted to know how many passengers on a flight were from Atlanta, GA? This is where joining is useful. The flight table does not contain the home address of the passenger but it does have a way to lookup that data using the name of the passenger.

Similarly, as a biological example, you might be a field biologist that has

1) a table of sites you are observing. This includes information such as GPS coordinates, state, monthly average temperature, altitude, etc.

2) a table of species that you are interested in. This includes taxonomic information, average height and weight, brood size information, mating and social behavior, etc.

3) a table of observations. This includes information about the site where the observations were made along with the species that was observed along with the date/time this observation was made.

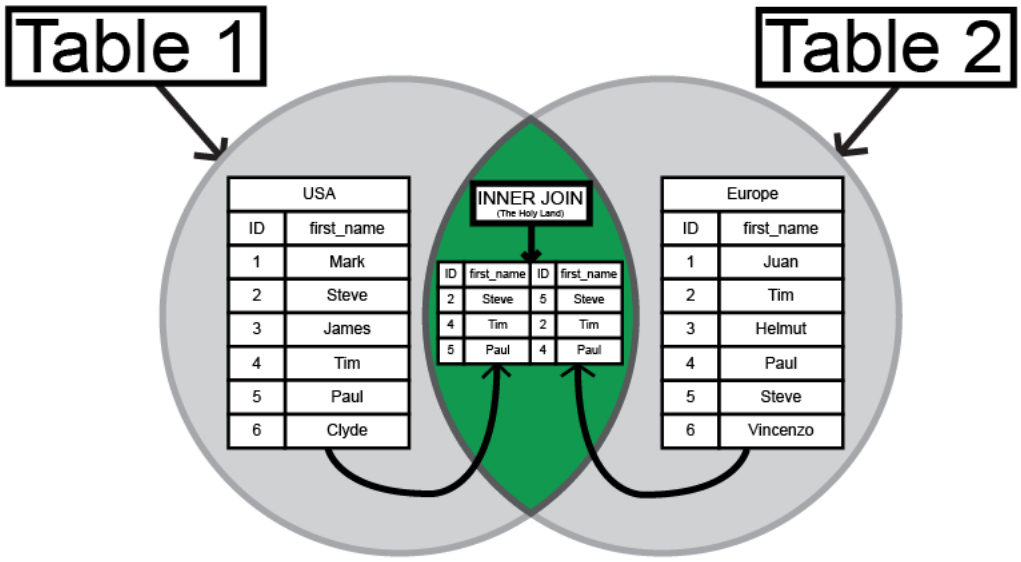
At the end of 5 years of research for your PhD, you might want to make a few analysis that require combining data from these three tables.

How often is a species observed as a function of altitude?

How often is a species observed as a function of temperature?

Are large species more likely to be observed during winter months than smaller months?

Joining is the process of combining data from two or more tables. Below shows how a join works using explicit data. Table 1 shows the 6 most common names in the US. Table 2 shows the 6 most common names in Europe. In order to merge two tables, you have to specify a column (or columns) to join on. In other words, if you want to combine data from one table with data from another table, you need to know which rows are the same in the table. In this case, if we joined on ID, we would match all 6 rows. If we joined on first_name, we would match three of the rows.



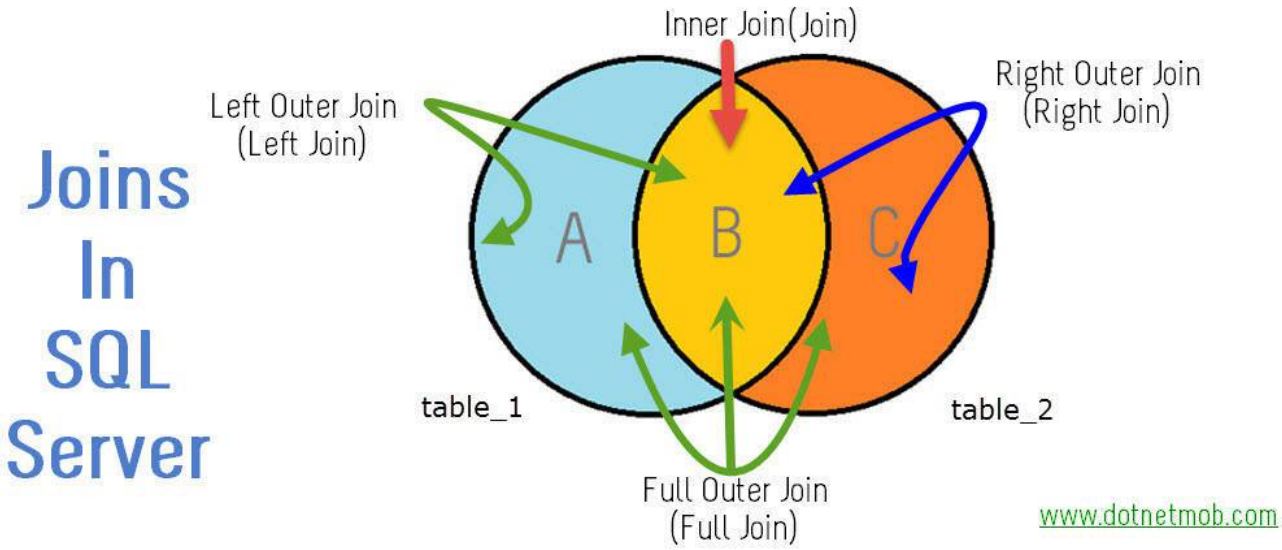
Types of joins

Inner join – Only include data records shared by both tables (3 rows would be returned in the above example)

Full outer join – Include all data records included in either table (9 rows would be returned in the above example)

Left outer join – Include all data records included in the left table (6 rows would be returned in the above example)

Right outer join – Include all data records included in the right table (6 rows would be returned in the above example)



Topic – Joining in pandas

To perform joins in pandas, you should use the merge function in pandas:

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.merge.html

As an example, let's read in two dataframes. The 1st is the population demographics of all the countries in the world. The 2nd is a dataframe I created (you should have created something similar in a previous exercise) that contains the IFR by age.

```
>>> import pandas as pd
>>> dt1 = pd.read_csv('WorldDemographics.csv', index_col = 0)
>>> dt2 = pd.read_csv('IFRxAge.csv', index_col = 0)
>>> dt1
      PopulationID  country_code    Level  continent_code  Age   #Alive
0      Afghanistan             4  Country            5501    0  1134501
1      Afghanistan             4  Country            5501    1  1134501
2      Afghanistan             4  Country            5501    2  1134501
3      Afghanistan             4  Country            5501    3  1124250
4      Afghanistan             4  Country            5501    4  1113998
...            ...           ...      ...             ...  ...      ...
20296     Zimbabwe           716  Country             910   96      163
20297     Zimbabwe           716  Country             910   97       53
20298     Zimbabwe           716  Country             910   98       41
20299     Zimbabwe           716  Country             910   99       30
20300     Zimbabwe           716  Country             910  100       19
```

```
[20301 rows x 6 columns]
>>> dt2
     Age      IFR
0      0  0.00002
1      1  0.00002
2      2  0.00002
3      3  0.00002
4      4  0.00002
..   ...      ...
96    96  0.09000
97    97  0.09000
98    98  0.09000
99    99  0.09000
100  100  0.09000

[101 rows x 2 columns]
```

As an example of why we might want to merge data from these two tables, we would like to estimate the number of people that will die by each age. To do this, we need data from the 1st table (e.g. the number of 50 year old people in Canada) and data from the 2nd table (the IFR for 50 year old people). We can group this by joining by age:

```
>>> dt3 = pd.merge(dt1, dt2, left_on = 'Age', right_on = 'Age', how = 'inner')
>>> dt3
              PopulationID  country_code    Level  continent_code  Age   #Alive      IFR
0              Afghanistan             4  Country            5501    0  1134501  0.00002
1                  Albania             8  Country             925    0    33255  0.00002
2                  Algeria            12  Country             912    0  1008303  0.00002
3                   Angola            24  Country             911    0  1159000  0.00002
4      Antigua and Barbuda            28  Country             915    0     1471  0.00002
...                    ...           ...      ...             ...  ...      ...      ...
20296              Viet Nam           704  Country             920  100    23850  0.09000
20297       Western Sahara            732  Country             912  100        1  0.09000
20298                Yemen           887  Country             922  100       78  0.09000
20299               Zambia           894  Country             910  100        5  0.09000
20300             Zimbabwe           716  Country             910  100       19  0.09000
```

Now we have a table that contains both data. Note that the order of the table has changed, sorted on Age. If we want the original order:

```
>>> dt3 = dt3.sort_values(['PopulationID','Age'])
```

Since the table has both the #Alive and the IFR for each row, it is now easy to calculate the predicted number of deaths:

```
>>> dt3['PredictedDeaths'] = dt3['#Alive']*dt3['IFR']
```

Now we can very easily calculate the predicted number of people that will die and the total IFR by population:

```
>>> dt3
      PopulationID  country_code    Level  continent_code  Age   #Alive      IFR  PredictedDeaths
0      Afghanistan             4  Country            5501    0  1134501  0.00002         22.69002
201    Afghanistan             4  Country            5501    1  1134501  0.00002         22.69002
402    Afghanistan             4  Country            5501    2  1134501  0.00002         22.69002
603    Afghanistan             4  Country            5501    3  1124250  0.00002         22.48500
804    Afghanistan             4  Country            5501    4  1113998  0.00002         22.27996
...            ...           ...      ...             ...  ...      ...      ...              ...
19496     Zimbabwe           716  Country             910   96      163  0.09000         14.67000
19697     Zimbabwe           716  Country             910   97       53  0.09000          4.77000
19898     Zimbabwe           716  Country             910   98       41  0.09000          3.69000
20099     Zimbabwe           716  Country             910   99       30  0.09000          2.70000
20300     Zimbabwe           716  Country             910  100       19  0.09000          1.71000

>>> dt4 = dt3.groupby('PopulationID').sum()[['#Alive','PredictedDeaths']]
>>> dt4
```

```
                        #Alive  PredictedDeaths
PopulationID
Afghanistan          38928204     117963.90656
Albania               2877640      42576.25422
Algeria              43850203     310409.63932
Angola               32866164      85247.39462
Antigua and Barbuda     97878        961.97790
...                       ...              ...
Viet Nam             97342743     820274.16220
Western Sahara         597282       2478.18834
Yemen                29825815      98839.72874
Zambia               18383891      46098.90062
Zimbabwe             14862850      49867.74150


>>> dt4['PredictedIFR'] = dt4['PredictedDeaths']/dt4['#Alive']
>>> dt4
                        #Alive  PredictedDeaths  PredictedIFR
PopulationID
Afghanistan          38928204     117963.90656      0.003030
Albania               2877640      42576.25422      0.014796
Algeria              43850203     310409.63932      0.007079
Angola               32866164      85247.39462      0.002594
Antigua and Barbuda     97878        961.97790      0.009828
...                       ...              ...           ...
Viet Nam             97342743     820274.16220      0.008427
Western Sahara         597282       2478.18834      0.004149
Yemen                29825815      98839.72874      0.003314
Zambia               18383891      46098.90062      0.002508
Zimbabwe             14862850      49867.74150      0.003355
```