# 04-Combining information in Pandas

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Often information is comming from different sources and it is necessary to combine it into one object. We are going to see the different ways in which information contained within separate Dataframes can be combined in a meaningful way.

## 4.1 Concatenation

The simplest way we can combine two Dataframes is simply to "paste" them together:

In [2]:
```
composers1 = pd.read_excel('Datasets/composers.xlsx', index_col='compose
r',sheet_name='Sheet1')
composers1
```

Out[2]:

| composer | birth | death | city |
|---|---|---|---|
| **Mahler** | 1860 | 1911 | Kaliste |
| **Beethoven** | 1770 | 1827 | Bonn |
| **Puccini** | 1858 | 1924 | Lucques |
| **Shostakovich** | 1906 | 1975 | Saint-Petersburg |

In [3]:
```
composers2 = pd.read_excel('Datasets/composers.xlsx', index_col='compose
r',sheet_name='Sheet3')
composers2
```

Out[3]:

| composer | birth | death | city |
|---|---|---|---|
| **Verdi** | 1813 | 1901 | Roncole |
| **Dvorak** | 1841 | 1904 | Nelahozeves |
| **Schumann** | 1810 | 1856 | Zwickau |
| **Stravinsky** | 1882 | 1971 | Oranienbaum |
| **Mahler** | 1860 | 1911 | Kaliste |

To be concatenated, Dataframes need to be provided as a list:

In [4]:
```
all_composers = pd.concat([composers1,composers2])
# columns have the same -> ok
```

In [5]:
```
all_composers
```

Out[5]:

| composer | birth | death | city |
|---|---|---|---|
| **Mahler** | 1860 | 1911 | Kaliste |
| **Beethoven** | 1770 | 1827 | Bonn |
| **Puccini** | 1858 | 1924 | Lucques |
| **Shostakovich** | 1906 | 1975 | Saint-Petersburg |
| **Verdi** | 1813 | 1901 | Roncole |
| **Dvorak** | 1841 | 1904 | Nelahozeves |
| **Schumann** | 1810 | 1856 | Zwickau |
| **Stravinsky** | 1882 | 1971 | Oranienbaum |
| **Mahler** | 1860 | 1911 | Kaliste |

One potential problem is that two tables contain duplicated information:

In [6]:
```
# be careful if same index !! can have twice the same index !
# ! ensure to not have the same index
all_composers.loc['Mahler']
```

Out[6]:

| composer | birth | death | city |
|---|---|---|---|
| **Mahler** | 1860 | 1911 | Kaliste |
| **Mahler** | 1860 | 1911 | Kaliste |

It is very easy to get rid of it using:

In [7]:
```
all_composers.drop_duplicates()
# suppress duplicates from the table
```

Out[7]:

| composer | birth | death | city |
|---|---|---|---|
| **Mahler** | 1860 | 1911 | Kaliste |
| **Beethoven** | 1770 | 1827 | Bonn |
| **Puccini** | 1858 | 1924 | Lucques |
| **Shostakovich** | 1906 | 1975 | Saint-Petersburg |
| **Verdi** | 1813 | 1901 | Roncole |
| **Dvorak** | 1841 | 1904 | Nelahozeves |
| **Schumann** | 1810 | 1856 | Zwickau |
| **Stravinsky** | 1882 | 1971 | Oranienbaum |

## 4.2 Joining two tables

An other classical case is that of two list with similar index but containing different information, e.g.

In [8]:
```
composers1 = pd.read_excel('Datasets/composers.xlsx', index_col='compose
r',sheet_name='Sheet1')
composers1
```

Out[8]:

| composer | birth | death | city |
|---|---|---|---|
| **Mahler** | 1860 | 1911 | Kaliste |
| **Beethoven** | 1770 | 1827 | Bonn |
| **Puccini** | 1858 | 1924 | Lucques |
| **Shostakovich** | 1906 | 1975 | Saint-Petersburg |

In [9]:
```
composers2 = pd.read_excel('Datasets/composers.xlsx', index_col='compose
r',sheet_name='Sheet4')
composers2
```

Out[9]:

| composer | first name |
|---|---|
| **Mahler** | Gustav |
| **Beethoven** | Ludwig van |
| **Puccini** | Giacomo |
| **Brahms** | Johannes |

If we we use again simple concatenation, this doesn't help us much. We just end up with a large matrix with lots of NaN's:

In [10]:
```
pd.concat([composers1, composers2])
```

```
/usr/local/lib/python3.5/dist-packages/ipykernel_launcher.py:1: FutureWar
ning: Sorting because non-concatenation axis is not aligned. A future ver
sion
of pandas will change to not sort by default.

To accept the future behavior, pass 'sort=False'.

To retain the current behavior and silence the warning, pass 'sort=True'.

  """Entry point for launching an IPython kernel.
```

Out[10]:

| | birth | city | death | first name |
|---|---|---|---|---|
| **composer** | | | | |
| **Mahler** | 1860.0 | Kaliste | 1911.0 | NaN |
| **Beethoven** | 1770.0 | Bonn | 1827.0 | NaN |
| **Puccini** | 1858.0 | Lucques | 1924.0 | NaN |
| **Shostakovich** | 1906.0 | Saint-Petersburg | 1975.0 | NaN |
| **Mahler** | NaN | NaN | NaN | Gustav |
| **Beethoven** | NaN | NaN | NaN | Ludwig van |
| **Puccini** | NaN | NaN | NaN | Giacomo |
| **Brahms** | NaN | NaN | NaN | Johannes |

The better way of doing this is to **join** the tables. This is a classical database concept avaialble in Pandas.

`join()` operates on two tables: the first one is the "left" table which uses `join()` as a method. The other table is the "right" one.

Let's try the default join settings:

In [11]:
```
composers1.join(composers2)
# get all elements of the 1st table, merged with the 2nd table
# everything based on the left table (what from the 2nd table and is not
in the 1st table is dropped)
# by default is left based
```

Out[11]:

| | birth | death | city | first name |
|---|---|---|---|---|
| **composer** | | | | |
| **Mahler** | 1860 | 1911 | Kaliste | Gustav |
| **Beethoven** | 1770 | 1827 | Bonn | Ludwig van |
| **Puccini** | 1858 | 1924 | Lucques | Giacomo |
| **Shostakovich** | 1906 | 1975 | Saint-Petersburg | NaN |

We see that Pandas was smart enough to notice that the two tables had a index name and used it to combine the tables. We also see that one element from the second table (Brahms) is missing. The reason for this is the way indices not present in both tables are handled. There are four ways of doing this with two tables called here the "left" and "right" table.

### 4.2.1. Join left

Here "left" and "right" just represent two Dataframes that should be merged. They have a common index, but not necessarily the same items. For example here Shostakovich is missing in the second table, while Brahms is missing in the first one. When using the "right" join, we use the first Dataframe as basis and only use the indices that appear there.

In [12]:
```
composers1.join(composers2, how = 'left') # this is the default
```

Out[12]:

| | birth | death | city | first name |
|---|---|---|---|---|
| **composer** | | | | |
| **Mahler** | 1860 | 1911 | Kaliste | Gustav |
| **Beethoven** | 1770 | 1827 | Bonn | Ludwig van |
| **Puccini** | 1858 | 1924 | Lucques | Giacomo |
| **Shostakovich** | 1906 | 1975 | Saint-Petersburg | NaN |

Hence Brahms is left out.

### 4.2.2. Join right

We can do the the opposite and use the indices of the second Dataframe as basis:

In [13]:
```
composers1.join(composers2, how = 'right')
```

Out[13]:

| | birth | death | city | first name |
|---|---|---|---|---|
| **composer** | | | | |
| **Mahler** | 1860.0 | 1911.0 | Kaliste | Gustav |
| **Beethoven** | 1770.0 | 1827.0 | Bonn | Ludwig van |
| **Puccini** | 1858.0 | 1924.0 | Lucques | Giacomo |
| **Brahms** | NaN | NaN | NaN | Johannes |

Here we have Brahms but not Shostakovich.

### 4.2.3. Inner, outer

Finally, we can just say that we want to recover eihter only the items that appaer in both Dataframes (inner, like in a Venn diagram) or all the items (outer).

In [14]:
```
composers1.join(composers2, how = 'inner')
# => take all indices of both tables
```

Out[14]:

| | birth | death | city | first name |
|---|---|---|---|---|
| **composer** | | | | |
| **Mahler** | 1860 | 1911 | Kaliste | Gustav |
| **Beethoven** | 1770 | 1827 | Bonn | Ludwig van |
| **Puccini** | 1858 | 1924 | Lucques | Giacomo |

In [15]:
```
composers1.join(composers2, how = 'outer')
```

Out[15]:

| | birth | death | city | first name |
|---|---|---|---|---|
| **composer** | | | | |
| **Beethoven** | 1770.0 | 1827.0 | Bonn | Ludwig van |
| **Brahms** | NaN | NaN | NaN | Johannes |
| **Mahler** | 1860.0 | 1911.0 | Kaliste | Gustav |
| **Puccini** | 1858.0 | 1924.0 | Lucques | Giacomo |
| **Shostakovich** | 1906.0 | 1975.0 | Saint-Petersburg | NaN |

### 4.3.4 Joining on columns : merge

Above we have used `join` to join based on indices. However sometimes tables don't have the same indices but similar contents that we want to merge. For example let's imagine whe have the two Dataframes below:

In [16]:
```
composers1 = pd.read_excel('Datasets/composers.xlsx', sheet_name='Sheet1
')
composers2 = pd.read_excel('Datasets/composers.xlsx', sheet_name='Sheet6
')
```

In [17]:
```
composers1
```

Out[17]:

| | composer | birth | death | city |
|---|---|---|---|---|
| **0** | Mahler | 1860 | 1911 | Kaliste |
| **1** | Beethoven | 1770 | 1827 | Bonn |
| **2** | Puccini | 1858 | 1924 | Lucques |
| **3** | Shostakovich | 1906 | 1975 | Saint-Petersburg |

In [18]:
```
composers2
```

Out[18]:

| | last name | first name |
|---|---|---|
| **0** | Puccini | Giacomo |
| **1** | Beethoven | Ludwig van |
| **2** | Brahms | Johannes |
| **3** | Mahler | Gustav |

The indices don't match and are not the composer name. In addition the columns containing the composer names have different labels. Here we can use `merge()` and specify which columns we want to use for merging, and what type of merging we need (inner, left etc.)

In [19]:
```
# take left and right tables and can specify which column from each to p
erform the merge
pd.merge(composers1, composers2, left_on='composer', right_on='last name
')
```

Out[19]:

| | composer | birth | death | city | last name | first name |
|---|---|---|---|---|---|---|
| **0** | Mahler | 1860 | 1911 | Kaliste | Mahler | Gustav |
| **1** | Beethoven | 1770 | 1827 | Bonn | Beethoven | Ludwig van |
| **2** | Puccini | 1858 | 1924 | Lucques | Puccini | Giacomo |

Again we can use another variety of join than the default inner:

In [20]:
```
pd.merge(composers1, composers2, left_on='composer', right_on='last name
',how = 'outer')
```

Out[20]:

| | composer | birth | death | city | last name | first name |
|---|---|---|---|---|---|---|
| **0** | Mahler | 1860.0 | 1911.0 | Kaliste | Mahler | Gustav |
| **1** | Beethoven | 1770.0 | 1827.0 | Bonn | Beethoven | Ludwig van |
| **2** | Puccini | 1858.0 | 1924.0 | Lucques | Puccini | Giacomo |
| **3** | Shostakovich | 1906.0 | 1975.0 | Saint-Petersburg | NaN | NaN |
| **4** | NaN | NaN | NaN | NaN | Brahms | Johannes |

In [21]:
```
pd.merge(composers1, composers2, left_on='composer', right_on='last name
',how = 'right')
```

Out[21]:

| | composer | birth | death | city | last name | first name |
|---|---|---|---|---|---|---|
| **0** | Mahler | 1860.0 | 1911.0 | Kaliste | Mahler | Gustav |
| **1** | Beethoven | 1770.0 | 1827.0 | Bonn | Beethoven | Ludwig van |
| **2** | Puccini | 1858.0 | 1924.0 | Lucques | Puccini | Giacomo |
| **3** | NaN | NaN | NaN | NaN | Brahms | Johannes |

In [22]:
```
# MZ: to remove the column that now contains duplicated information:
dt1 = pd.merge(composers1, composers2, left_on='composer', right_on='las
t name')
dt1
# default is to drop rows, to drop columns set 1st axis
dt1.drop('last name', axis=1)
dt1
dt1.drop('last name', axis=1, inplace=True)
dt1
```

Out[22]:

| | composer | birth | death | city | first name |
|---|---|---|---|---|---|
| **0** | Mahler | 1860 | 1911 | Kaliste | Gustav |
| **1** | Beethoven | 1770 | 1827 | Bonn | Ludwig van |
| **2** | Puccini | 1858 | 1924 | Lucques | Giacomo |