**Computing multiple aggregates of multiple columns**

The .agg() method can be used with a tuple or list of aggregations as input. When applying multiple aggregations on multiple columns, the aggregated DataFrame has a *multi-level column index*.

In this exercise, you're going to group passengers on the Titanic by 'pclass' and aggregate the 'age' and 'fare' columns by the functions 'max' and 'median'. You'll then use multi-level selection to find the oldest passenger per class and the median fare price per class.

The DataFrame has been pre-loaded as titanic.

**INSTRUCTIONS**

* Group titanic by 'pclass' and save the result as by\_class.
* Select the 'age' and 'fare' columns from by\_class and save the result as by\_class\_sub.
* Aggregate by\_class\_sub using 'max' and 'median'. You'll have to pass 'max' and 'median' in the form of a list to .agg().
* Use .loc[] to print all of the rows and the column specification ('age','max'). This has been done for you.
* Use .loc[] to print all of the rows and the column specification ('fare','median').

# Group titanic by 'pclass': by\_class

print(titanic.head())

by\_class = titanic.groupby('pclass')

# Select 'age' and 'fare'

by\_class\_sub = by\_class[['age','fare']]

print(by\_class\_sub.head())

# Aggregate by\_class\_sub by 'max' and 'median': aggregated

aggregated = by\_class\_sub.agg(['max', 'median'])

# Print the maximum age in each class

print(aggregated.loc[:, ('age','max')])

# Print the median fare in each class

print(aggregated.loc[:, ('fare','median')])