df.shape display the size (x, y)

df.head() display the first n rows (default=5)

df.tail() display the last n rows (default=5)

df.describe() display the statistics of numerical data types

df.info() display the number of entries (rows), number of columns, and the data types

df.columns print a list of the columns

df['column\_name'] select a single column ( returns a Series)

df['column\_name'].value\_counts() count the number of object and boolean occurrences

df.sort\_values(by='column\_name') sort the values in the given column

df.drop() remove rows or columns by specifying the label or index of the row/column

df.isnull().sum() count and sum the number of null occurrences (NaN or None)

df.fillna() fill NaN values in a variety of ways

df.dropna() remove values that are NaN or None; by default removes all rows with NaNs

Some of the arguments are similar to what we have seen already with concatenation. But with pd.merge() we can do a lot of different types of joins. Let's summarize some of the different parameters.

left - one of the DataFrames to merge

right - the other DataFrame to merge

how - specifies the kind of merge to make

on - the column on which to merge; needs to be in both DataFrames

left\_on - the column to join on in the left DataFrame

right\_on - the column to join on in the right DataFrame

left\_index or right\_index - use the index from the left/right DataFrame as the join key(s)

pd.merge(how=’outer’) = UNION (Joins & displays NaNs) vs pd.merge(how=’inner’) = INTERSECTION (Joins & displays only rows with data in all 3 columns)