```
25%    40.500000  39.500000  22.000000  15.500000
50%    47.000000  47.000000  44.000000  22.000000
75%    67.500000  63.500000  62.500000  36.500000
max    74.000000  73.000000  66.000000  52.000000
```

## Correlation

Correlation shows how much relationship exists between two variables. Parametric machine learning methods such as logistic and linear regression can take a performance hit when variables are highly correlated. The correlation values range from –1 to 1, with 0 indicating no correlation at all. –1 signifies that the variables are strongly negatively correlated, while 1 shows that the variables are strongly positively correlated. In practice, it is safe to eliminate variables that have a correlation value greater than –0.7 or 0.7. A common correlation estimate in use is the Pearson's correlation coefficient.

```
my_DF.corr(method='pearson')
'Output':
           First    Second     Third    Fourth
First    1.000000  0.587645 -0.014100 -0.317333
Second   0.587645  1.000000 -0.768495 -0.345265
Third   -0.014100 -0.768495  1.000000  0.334169
Fourth  -0.317333 -0.345265  0.334169  1.000000
```

## Skewness

Another important statistical metric is the skewness of the dataset. Skewness is when a bell-shaped or normal distribution is shifted toward the right or the left. Pandas offers a convenient function called **skew()** to check the skewness of each variable. Values close to 0 are more normally distributed with less skew.

```
my_DF.skew()
'Output':
First    -0.167782
Second   -0.566914
Third    -0.084490
Fourth    0.691332
dtype: float64
```

# Importing Data

Again, getting data into the programming environment for analysis is a fundamental and first step for any data analytics or machine learning task. In practice, data usually comes in a comma-separated value, **csv,** format.

```
my_DF = pd.read_csv('link_to_file/csv_file', sep=',', header = None)
```

To export a DataFrame back to **csv**

```
my_DF.to_csv('file_name.csv')
```

For the next example, the dataset 'states.csv' is found in the chapter folder of the code repository of this book.

```
my_DF = pd.read_csv('states.csv', sep=',', header = 0)

# read the top 5 rows
my_DF.head()

# save DataFrame to csv
my_DF.to_csv('save_states.csv')
```

# Timeseries with Pandas

One of the core strengths of Pandas is its powerful set of functions for manipulating timeseries datasets. A couple of these functions are covered in this material.

## Importing a Dataset with a DateTime Column

When importing a dataset that has a column containing datetime entries, Pandas has an attribute in the **read_csv** method called **parse_dates** that converts the datetime column from strings into Pandas **date** datatype. The attribute **index_col** uses the column of datetimes as an index to the DataFrame.

The method **head()** prints out the first five rows of the DataFrame, while the method **tail()** prints out the last five rows of the DataFrame. This function is very useful for taking a peek at a large DataFrame without having to bear the computational cost of printing it out entirely.