4.4.2

Doing a Big-Picture Analysis

Before diving into the details of our data, it's helpful to get a sense of its overarching characteristics. So, we do a big-picture analysis of the data. This might focus on the number of elements in the dataset (the count); the minimum, maximum, and mean (average) values; and how spread out the data is around that mean value (the standard deviation). Luckily, Pandas offers several functions that help establish this big-picture overview of a dataset. Let's learn how to use them.

Generate a Bunch of Summary Statistics

One of the most important steps in any analysis is to summarize your data. **Summarization** is an analysis technique that uses statistical measurements to describe the values in a dataset. **Summary statistics**, which include the count, mean, standard deviation, and minimum and maximum values, supply general information about the values in a dataset. With this information, you can build an intuition about your data. That is, you can make sense of it and compare it to other datasets.

The Pandas `describe` function offers the most straightforward way to generate a bunch of summary statistics all at once.

For example, consider a DataFrame named `sales_df` that has three columns—"item", "quantity", and "total_price"—as the following code shows:

```
sales_df = pd.DataFrame({"item": ["tomatoes", "onions", "potatoes", "tomatoes", "mushrooms"],
                "quantity": [15, 26, 10, 12, 2],
                "total_price": [50.25, 39.52, 50.01, 40.20, 8.46]})
sales_df.head()
```

The DataFrame has five rows of produce sales data, as the following image shows:

```
[2]: sales_df = pd.DataFrame({"item": ["tomatoes", "onions", "potatoes", "tomatoes", "mushrooms"],
                              "quantity": [15, 26, 10, 12, 2],
                              "total_price": [50.25, 39.52, 50.01, 40.20, 8.46]})
     sales_df.head()
```

[2]:

	item	quantity	total_price
0	tomatoes	15	50.25
1	onions	26	39.52
2	potatoes	10	50.01
3	tomatoes	12	40.20
4	mushrooms	2	8.46

To call the `describe` function on this DataFrame, we use the following code:

```
sales_df.describe()
```

Running the preceding code produces the output that the following image shows:

```
: sales_df.describe()
```

:

	quantity	total_price
count	5.000000	5.000000
mean	13.000000	37.688000
std	8.717798	17.128761
min	2.000000	8.460000
25%	10.000000	39.520000
50%	12.000000	40.200000
75%	15.000000	50.010000
max	26.000000	50.250000

As the preceding image shows, the output consists of the summary statistics for each column of numeric data in the DataFrame. In this case, these are the "quantity", and "total_price" columns.

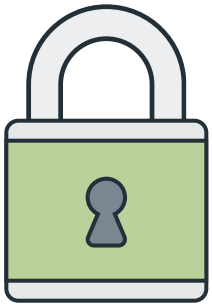
IMPORTANT

The Pandas `describe` function always outputs the same list of summary statistics:

- count: The number of elements in the column.
- mean: The average value of the column.
- std: The standard deviation of the values in the column—that is, how spread out the values are (which indicates the distribution).
- min: The minimum value in the column.
- 25%, 50%, 75%: The values that occupy each respective percentile (which also indicate the distribution). Note that the value that occupies the 50% percentile isn't the same as the mean, although it's close.
- max: The maximum value in the column.

By default, the `describe` function generates summary statistics for those columns that contain only numeric content. To generate summary statistics for columns that contain mixed data types, include the `include='all'` parameter—for example, `df.describe(include='all')`.

Now, before moving on, check your knowledge in the following assessment:



Access Denied

You don't have access to view this resource.

Generate a Specific Summary Statistic

Pandas also has functions that we can use to generate specific summary statistics, including `mean`, `min`, and `max`.

The following code generates the mean of every numeric column in our example DataFrame:

```
sales_df.mean()
```

```
sales_df.mean()

quantity       13.000
total_price    37.688
dtype: float64
```

This following code retrieves the maximum value of every column, including non-numeric columns:

```
sales_df.max()
```

```
sales_df.max()

item            tomatoes
quantity               26
total_price         50.25
dtype: object
```

IMPORTANT

The Pandas `min` and `max` functions generate the minimum and the maximum value, respectively, of every column. These values will likely come from different rows.

We can also apply these functions to individual columns.

For example, the following code generates the minimum value of just the "total_price" column and stores the returned value in `min_price`:

```
min_price = sales_df["total_price"].min()
min_price
```

Running the preceding code stores 8.46 in `min_price`. That's the smallest value in the "total_price" column. It might prove useful to generate the whole row for the minimum price. That way, we can discover which product had the smallest sale. We'll learn how to do that in a later lesson.

Now that you've learned about doing a big-picture analysis, you'll next have the opportunity to do so on your own in the following activity.