4.5.3   Determine Data Types

You have been tasked with getting key metrics from the datasets. This will involve performing calculations to get sums, averages, and percentages. Before you perform these calculations, though, you should check if the numbers in these datasets are the correct data type for making those calculations, as this is considered a best practice.

Here are six common data types that we may encounter in this module and that you may come across in the future:

Data Type	Pandas Name	Example
Boolean	`bool`	"True" and "False"
Integer	`int32`	Values from − 2,147,483,648 to 2,147,483,647

Integer	`int64`	Values from −9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
Float	`float64`	Floating decimal
Object	`O, object`	Typically strings; often used as a catchall for columns with different data types or other Python objects like tuples, lists, and dictionaries
Datetime	`datetime64`	Specific moment in time with nanosecond precision, i.e., 2019-06-03 16:04:00.465107

REWIND

- If integers store values from −32,768 to 32,767, they are 16 bits or `int16`.

- If integer store values from −2,147,483,648 to 2,147,483,647, they are 32 bits or `int32`.

- If integer store values from

- −9,223,372,036,854,775,808 to 9,223,372,036,854,775,807, they are 64 bits, or `int64`.

With the Pandas library, we can check the data types of each column by using the Pandas `dtypes` attribute on a DataFrame.

To determine the data types in `school_data.csv`, type and run the following code:

```
# Determine data types for the school DataFrame.
school_data_df.dtypes
```

This code returns a Series with the name of each column and the data type of the values in the column. As shown in the output below, "School ID," "size," and "budget" are integers, which is the correct data type for performing arithmetic calculations.

```
School ID          int64
school_name        object
type               object
size               int64
budget             int64
dtype: object
```

We can also use the `dtype` attribute on a single column. There are two ways to perform this operation. We can use `df.column.dtype` if the column name doesn't have any spaces in the name, like "school_name". If there are spaces, we must use `df["column"].dtype`.

 ↻ Retake

Next, determine the data types for the student data file. Type and run the following code:

```
# Determine data types for the student DataFrame.
student_data_df.dtypes
```

The code returns each column name and the data type of each column. As shown in the output below, "Student ID," "reading_score," and "math_score" are integers, which is the correct data type for performing arithmetic calculations.

```
Student ID              int64
student_name            object
gender                  object
grade                   object
school_name             object
reading_score           int64
math_score              int64
dtype: object
```

Based on the output, we determined that all of the columns we need to use for calculations are integers. Therefore, we won't need to change the data types for these columns. However, there may be instances in which it's necessary to change the data type. Some CSV and text files, for example, may contain numbers as strings (or objects) rather than integers. These numbers would need to be converted to integers or floats.

NOTE

For more information, see the Pandas documentation on the following topics:

- **Get the data types of a DataFrame (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.dtypes.html)**

- **Changing data types   (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.astype.html)**