Part I

Core pandas

Chapter 1

The DataFrame object

1.1 Selecting rows from a DataFrame

1.1.1 Extracting rows by index label

We can use loc to extract a sequence of index labels. The syntax mirrors Python's list slicing syntax. We provide the starting value, a colon, and the ending value. For extractions like this one, I strongly recommended sorting the index first, as it accelerates the speed with which pandas finds the value.

1.1.2 Extracting rows by index position

1.1.3 Extracting values from specific columns

We can use two alternative attributes, at and iat, when we know that we want to extract a single value from a DataFrame. The two attributes are speedier because pandas can optimize its searching algorithms when looking for a single value.

1.2 Resetting an index

Chapter 2

Filtering a DataFrame

2.1 Optimizing a data set for memory use

2.1.1 Converting data types with the astype method

The astype method converts a Series' values to a different data type. It accepts a single argument: the new data type. We can pass either the data type or a string with its name.

Updating a DataFrame column works similarly to setting a key-value pair in a dictionary. If a column with the specified name exists, pandas overwrites it with the new Series. If the column with the name does not exist, pandas creates a new Series and appends it to the right of the DataFrame. The library matches rows in the Series and DataFrame by shared index labels.

2.2 Filtering by a single condition

As long as you provide a Boolean Series, pandas will be able to filter the DataFrame.

2.3 Filtering by multiple conditions

We can filter a DataFrame with multiple conditions by creating two independent Boolean Series and then declaring the logical criterion that pandas should apply between them.

2.3.1 The AND condition

2.3.2 The OR condition

2.3.3 Inversion with ∼

2.3.4 Methods for Booleans

2.4 Filtering by condition

2.4.1 The isin method

The isin method, which accepts an iterable of elements (list, tuple, Series, and so on) and returns a Boolean Series. True denotes that pandas found the row's value among the iterable's values, and False denotes that it did not.

An optimal situation for using the isin method is when we do not know the comparison collection in advance, such as when it is generated dynamically.

2.4.2 The between method

Note that the first argument, the lower bound, is inclusive, and the second argument, the upper bound, is exclusive.

2.4.3 The isnull and notnull methods

The isnull and notnull methods are the best way to quickly filter for present and missing values in one or more rows.

2.4.4 Dealing with null values

We can use the subset parameter to target rows with a missing value in a specific column.

2.5 Dealing with duplicates

2.5.1 The duplicated method

The duplicated method's keep parameter informs pandas which duplicate occurrence to keep. Its default argument, "first", keeps the first occurrence of each duplicate value. We can also ask pandas to mark the last occurrence of a value in a column as the nonduplicate. Pass a string of "last" to the keep parameter.

2.5.2 The drop_duplicates method

We can pass the method a subset parameter with a list of columns that pandas should use to determine a row's uniqueness.

One additional option is available for the keep parameter. We can pass an argument of False to exclude all rows with duplicate values. Pandas will reject a row if there are any other rows with the same value.

Part II

Applied pandas

Chapter 3

Working with text data

3.1 Letter casing and whitespace

The Series object's str attribute exposes a StringMethods object, a powerful toolbox of methods for working with strings

3.2 String slicing

3.3 String slicing and character replacement

3.4 Boolean methods

The contains method checks for a substring's inclusion in each Series value. The method returns True when pandas finds the method's argument within the row's string and False when it does not.

3.5 Splitting strings

To ensure an equal number of elements per list, we can limit the number of splits. If we set a maximum threshold of one split, pandas will split a string at the first space and stop. Then we'll have a Series consisting of two-element lists. Each list will hold the customer's first name and anything that follows it.

We can use str.get to pull out a value from each row's list based on its index position.

3.6

3.7

Chapter 4

MultiIndex DataFrames

4.1 The MultiIndex object

To summarize, a MultiIndex is a storage container in which each label holds multiple values. A level consists of the values at the same position across the labels.

4.2 MultiIndex DataFrames

4.3 Sorting a MultiIndex

Pandas can find a value in an ordered collection much quicker than in a jumbled one. A good analogous example is searching for a word in a dictionary. It's easier to locate a word when words are in alphabetical order rather than a random sequence. Thus, **it's optimal to sort an index before selecting any rows and columns from a DataFrame**.

When we invoke the method on a MultiIndex DataFrame, pandas sorts all levels in ascending order and proceeds from the outside in.

4.4 Selecting with a MultiIndex

4.4.1 Extracting one or more columns

If we pass a single value in square brackets, pandas will look for it in the outermost level of the columns' MultiIndex.

To specify values across multiple levels in the column's MultiIndex, we can pass them inside a tuple.

4.5

4.6