

16.4.3

Spark Functions

Working with Spark DataFrames seems simple enough. Now you'll run some of PySpark's functions and actually analyze your data.

In the last section, you might have noticed that when calling a column with Spark, no real results were shown. As you might recall, Spark uses lazy evaluation. This means that Spark takes a list of instructions and formulates the best way to fulfill them, and then waits until you tell Spark to complete them. The process of listing and reading the instructions is called **transformation**, and your directive to complete them is called an **action**.

For example, say you have a new coffee maker. First, you read the instructions to learn what you need to do. You don't immediately make coffee, though. Instead, you wait for someone to ask if you can make coffee. Once you receive that request, you follow all the steps to make coffee. The transformation is you reading the instructions, and the action is the request to make coffee.

Let's start with an example.

NOTE

If you're using a new notebook for the next example, be sure to install Spark in the first cell, as we did in the previous example.

Start by creating a Spark session and loading in data, using the following code:

```
# Start Spark session
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("DataFrameFunctions").getOrCreate()
```

```
# Read in data from S3 Buckets
from pyspark import SparkFiles
url = "https://s3.amazonaws.com/dataviz-curriculum/day_1/wine.csv"
spark.sparkContext.addFile(url)
df = spark.read.csv(SparkFiles.get("wine.csv"), sep=",", header=True)

# Show DataFrame
df.show()
```

Now we'll use this data to identify the difference between transformations and actions.

Transformations

Transformations are the instructions for the computation. With the data loaded in, let's perform some transformations. What do you think will happen after we run the following code?



```
# Order a DataFrame by ascending values
df.orderBy(df["points"].desc())
```

If you guessed that nothing would happen yet, you would be correct. Here we applied the transformation to order the DataFrame by points in descending order. All we're doing is telling Spark that we want this DataFrame to be organized in this particular way, and Spark says, "Okay, got it—just let me know when you want me to do this."

Actions

Actions direct Spark to perform the computation instructions and return a result. What do you think will happen when you add `.show(5)` to the DataFrame?

```
df.orderBy(df["points"].desc()).show(5)
```

```
+-----+-----+-----+-----+-----+-----+
|country|description|designation|points|price|province|
+-----+-----+-----+-----+-----+-----+
|Italy|Here's a wow...|Messorio|99|320|Tusca
|US|A stupendous Pino...|Precious Mountain...|99|94|California|
|Italy|The 2007 Ornellai...|Ornellaia|99|200|Tuscany|
|US|Depth and texture...|Royal City Stoner...|99|140|Washington|
|France|A magnificent Cha...|Dom Prignon Oeno...|99|385|Champagne|
+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

If you guessed that a DataFrame organized by points would display, you would be correct. The `show(5)` method is an action that tells Spark to show the first five results.

Recap

Let's break down these concepts one more time:

- `orderBy()` and `desc()` are transformations telling Spark how to organize the data. Spark will read these transformations as instructions, but it won't act on them just yet.
- `show()` is an action that gives the go-ahead for Spark to run all of those transformations and to produce a result.

SKILL DRILL

Try to get the top 50 rows for values in ascending order on the points column.

More Functions

Let's take a look at a few more functions that we can use with Spark. As you continue to enter the code in your notebooks, keep in mind which methods you think are transformations and which ones are actions.

Spark can import additional functions, such as averages. Type and run the following code:

```
# Import functions
from pyspark.sql.functions import avg
df.select(avg("points")).show()
```

```
+-----+
|      avg points      |
+-----+
| 87.88834105383143   |
+-----+
```

The `avg()` function is the transformation, and `show()` is the action.

Spark can filter on columns by supplying the name of the column and operator and what to compare it against. Refer to the following code and results:

```
# Filter
df.filter("price<20").show(5)
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| country|description|designation|points|price| province| region_1| region_2| variety| winery|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| Bulgaria|This Bulgarian Ma...|Bergu| 90| 15| Bulgaria| null| null| Mavrud| Villa Melnik|
| Spain|Earthy plum and c...|Amandi| 90| 17| Galicia| Ribeira Sacra| null| Menc_ a| Don Bernardino|
| US|There's a lot to ...| null| 90| 18| California| Russian River Valley| Sonoma| Chardonnay| De Loach|
| US|Massively fruity,...| null| 91| 19| Oregon| Willamette Valley| Willamette Valley| Pinot Gris| Trinity Vineyards|
| Portugal|It is the ripe da...| Premium| 91| 15| Alentejo| null| null| Portuguese Red| Adega Cooperativa...|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

`Filter` is the transformation and `show` is the action.

We can also filter and select certain columns:

```
# Filter by price on certain columns
df.filter("price<20").select(['points', 'country', 'winery', 'price']).show(5)
```

```
+-----+-----+-----+-----+
| points| country| winery| price|
+-----+-----+-----+-----+
| 90| Bulgaria| Villa Melnik| 15|
| 90| Spain| Don Bernardino| 17|
| 90| US| De Loach| 18|
| 91| US| Trinity Vineyards| 19|
| 91| Portugal| Adega Cooperativa...| 15|
+-----+-----+-----+-----+
```

Both `filter` and `select` are separate transformations, and `show` is again the action.

Spark has multiple ways to perform transformations. For instance, how we were filtering our DataFrame was actually using SQL context with `price<20`.

However, we can also perform the same transformations using Python:

```
# Filter
df.filter("price<20").show(5)
# Filter by price on certain columns
df.filter("price<20").select(['points', 'country', 'winery', 'price']).show(5)

# Filter on exact state
df.filter(df["country"] == "US").show()
```

You might notice that there are more transformations than actions. Usually, we want to do several things with a dataset, but we'll only want to see those results in a few ways.

SKILL DRILL

Using both the SQL and Python context, use filtering to find the rows that contain a bottle of wine over \$15 and that comes from California.

Now that you have an understanding of using Spark, it's time to dive into the next step of your challenge: learning how computers interpret textual data. For this step we'll explore natural language processing, which we'll use in conjunction with Spark.

© 2020 - 2022 Trilogy Education Services, a 2U, Inc. brand. All Rights Reserved.