4.6.3

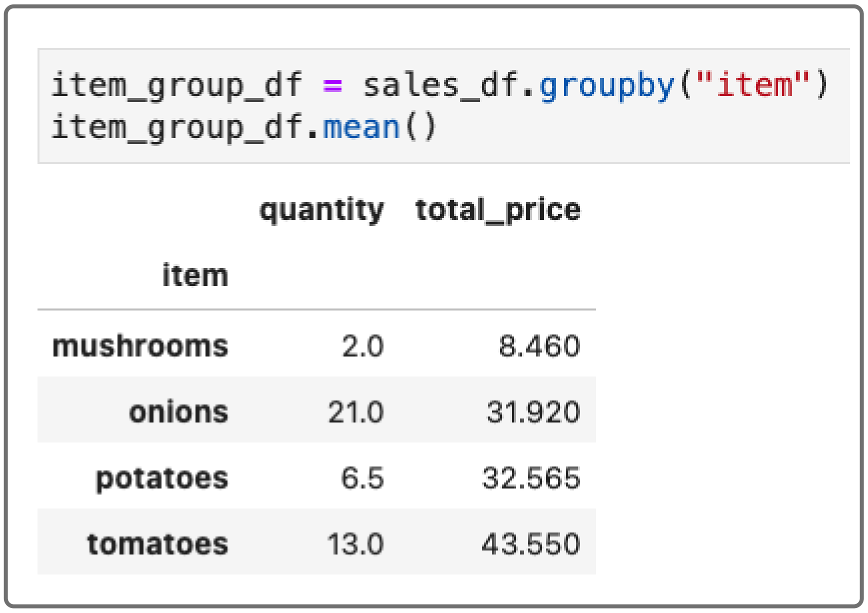
Compare the Data Aggregations

To compare the data aggregations, or groups, that we created by using the `groupby` function, we need to use a Pandas aggregation function. The **aggregation functions** are functions that each return a single value when applied to an entire column. For instance, `sum` adds all the values together to return a single value, while `max` finds and returns only the largest value.

For example, lets try to use `mean` on our grouped sales data, as the following code shows:

```
item_group_df = sales_df.groupby("item")
item_group_df.mean()
```

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



```
item_group_df = sales_df.groupby("item")
item_group_df.mean()
```

item	quantity	total_price
mushrooms	2.0	8.460
onions	21.0	31.920
potatoes	6.5	32.565
tomatoes	13.0	43.550

In the preceding image, notice that running `groupby` and then `mean` produced a DataFrame in which each unique value from the "item" column got assigned a row. And, the `mean` aggregation function placed the averages for these "item" groups separately in each column (that is, in the "quantity" and "total_price" columns).

IMPORTANT

The `groupby` function will show meaningful output only after an aggregation function is applied.

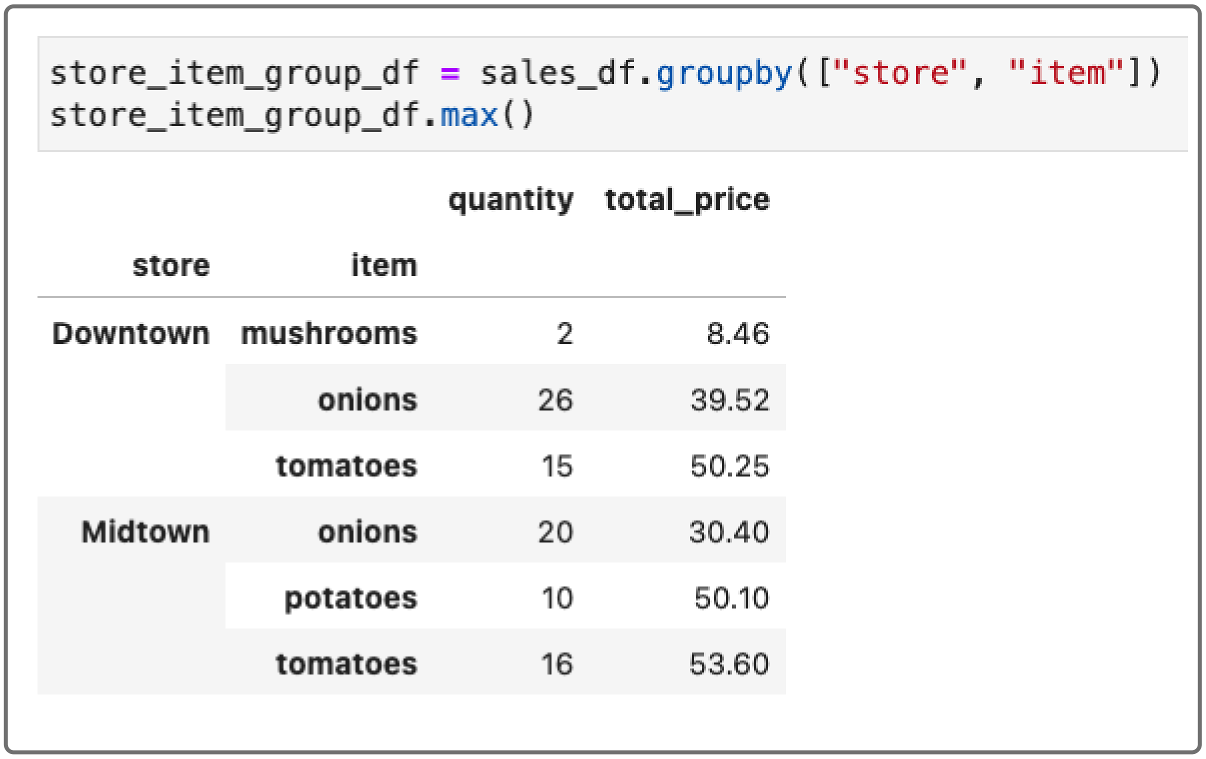
Now, we can compare various averages for each item! Using the output, we can observe that the average total price for a sale of tomatoes is $43.550 but only $31.920 for a sale of onions.

What if we want to compare the sales of particular items at different stores? Good news: we can group by multiple columns at the same time.

For example, the following code groups the sales data by store and then by item. Then, it displays the maximum values for each item in each store:

```
store_item_group_df = sales_df.groupby(["store", "item"])
store_item_group_df.max()
```

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



```
store_item_group_df = sales_df.groupby(["store", "item"])
store_item_group_df.max()
```

		quantity	total_price
store	**item**		
Downtown	**mushrooms**	2	8.46
	onions	26	39.52
	tomatoes	15	50.25
Midtown	**onions**	20	30.40
	potatoes	10	50.10
	tomatoes	16	53.60

In the preceding image, notice that the rows are grouped by store and then by item. And, notice that "mushrooms" appears only for the "Downtown" store, and "potatoes" appears only for the "Midtown" store. That's because the "Downtown" store sold no potatoes, and the "Midtown" store sold no mushrooms! Using the `groupby` function with multiple columns creates groups only for pairings that exist in the original DataFrame.

SHOW PRO TIP

We just learned that we can combine grouping with functions that we learned about in earlier lessons to help with an analysis. So with the combination of everything that we've learned, we can answer most questions about our data!

Now that you've learned how to compare the data aggregations, you'll next have the opportunity to do so yourself in the following activity.