We use
"DataFrame.value_counts"
to count the frequency of
each unique row in the
DataFrame.

Product
CPU
CPU
Monitor
Software
Monitor



Output:

Manager Product
Debra CPU 2
Fred Monitor 2
John Software 1



What Makes This Ebook Unique

Tips for Effective Use of This Ebook

Introduction

subset

normalize

sort

ascending

dropna

Contacts and Social Media

Sources & References

What Makes This Ebook Unique

- "The aim of this ebook is to give you the 'aha' moment right away at the start of learning a new concept."
- Practical Step By Step Guide With Simple Examples
- Visual Illustrations and Interactive
- Simple Datasets
- Comprehensive Coverage (pandas Documentation used as reference)

Tips for Effective Use of This Ebook

- Use Examples First: If you don't understand the text, go straight to the examples—they're selfexplanatory.
- After reviewing them, return to the text to grasp the practical concepts.
- Practice with Datasets: Copy the datasets provided and practice using them to reinforce your understanding.

Click on the blue links to go directly to the section you want to learn about.

Intoduction

We use

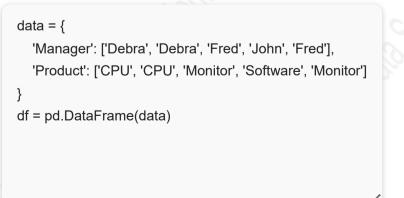
"DataFrame.value_counts" to count the frequency of each unique row in the DataFrame.

Product
CPU
CPU
Monitor
Software
Monitor

Output:

Manager Product
Debra CPU 2
Fred Monitor 2
John Software 1

Code Snippet



Manager	Product
Debra	CPU
Debra	CPU
Fred	Monitor
John	Software
Fred	Monitor

subset

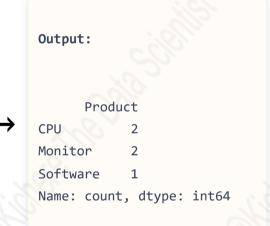
Accepts a label or a list of labels.

It specifies the columns to consider when counting unique combinations.

How many times does each product appear in the DataFrame, regardless of the manager?

The subset parameter with a single column like 'Product' returns the count of each product, regardless of manager.





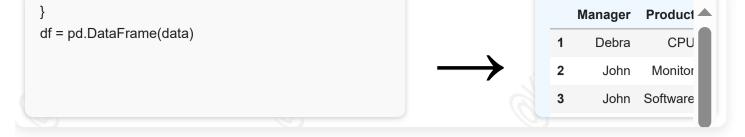
df.value_counts(
 subset=['Product']
)

Code:

Code Snippet

```
data = {
   'Manager': ['Debra', 'Debra', 'John', 'John', 'Fred'],
   'Product': ['CPU', 'CPU', 'Monitor', 'Software', 'Monitor']
```

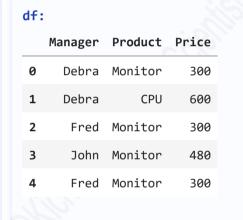
	Manager	Product	4
0	Debra	CPU	
			п



How many times does each unique combination of product and price appear in the DataFrame, disregarding the manager?

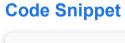
When subset accepts a list of column labels (like

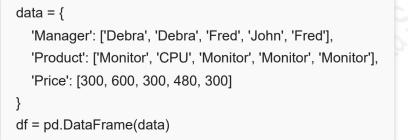
['Product', 'Price']), it returns counts of unique combinations of those columns, regardless of manager.









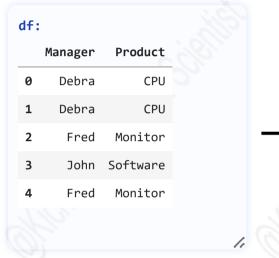


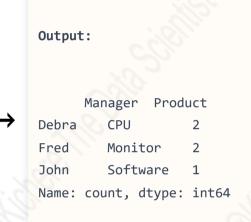
	Manager	Product	Price
0	Debra	Monitor	300
1	Debra	CPU	600
2	Fred	Monitor	300

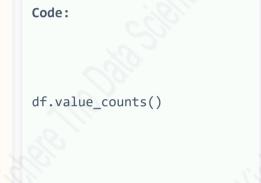


How many times does each combination of 'Manager' and 'Product' occur in the DataFrame?

Without specifying a subset, value_counts() counts unique combinations of all columns in the DataFrame.











	Manager	Product
0	Debra	CPU
1	Debra	CPU
2	Fred	Monitor
3	John	Software
4	Fred	Monitor

normalize

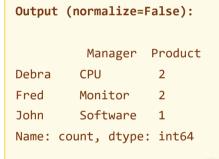
The **"normalize"** parameter returns proportions rather than frequencies. It accepts a **Boolean** value and defaults to **False**.

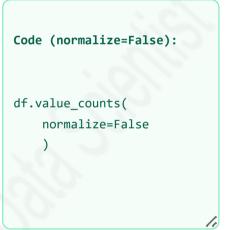
Count and Proportion of Unique Combinations of Manager and Product

(normalize=False):

Shows how often each unique row appears.

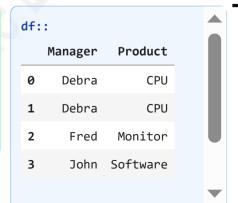
df:		
	Manager	Product
0	Debra	CPU
1	Debra	CPU
2	Fred	Monitor
3	John	Software
4	Fred	Monitor





(normalize=True):

Shows the proportion (percentage) of each unique row.



```
Output (normalize=True):

Manager Product
Debra CPU 0.4
Fred Monitor 0.4
John Software 0.2
Name: proportion, dtype:
```

```
Code (normalize=True):

df.value_counts(
    normalize=True
   )
```

floate

Code Snippet

```
data = {
   'Manager': ['Debra', 'Debra', 'Fred', 'John', 'Fred'],
   'Product': ['CPU', 'CPU', 'Monitor', 'Software', 'Monitor']
}
df = pd.DataFrame(data)
```

df:

Debra	CPU
Debra	CPU
Fred	Monitor
John	Software
Fred	Monitor
	Fred

sort

The **"sort"** parameter accepts a **Boolean** value and defaults to **True**.

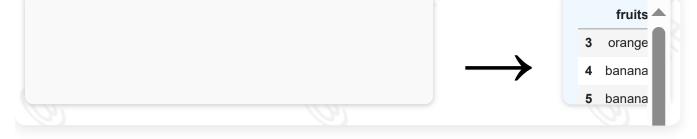
It sorts by frequencies when **True**.

It sorts by DataFrame column values when False.

value_counts(sort=True vs False) Behavior

```
df:
When sort=True
                                                 Output (sort=True):
                                                                          Code (sort=True):
                             fruits
(default):
                             apple
It sorts by frequencies
                                                           fruits
                         1 banana
                                                                          df.value counts(
                                                           3
(most frequent first).
                                                 banana
                                                 apple
                                                                              sort=True
                                                           2
                              apple
                                                 orange
                                                           1
                          3 orange
                                                 Name: count, dtype:
                                                                          # or simply:
                                                 int64
                                                                          df.value_counts()
                          4 banana
                          5 banana
When sort=False:
                                                 Output (sort=False):
                                                                          Code (sort=False):
                         df::
Sorts by DataFrame
                             fruits
column values.
                                                           fruits
                             apple
                                                                          df.value counts(
                                                 apple
                                                           2
                                                                              sort=False
                          1 banana
                                                 banana
                                                           3
                                                 orange
                                                           1
                             apple
                                                 Name: count, dtype:
                          3 orange
                                                 int64
                          4 banana
                          5 banana
```

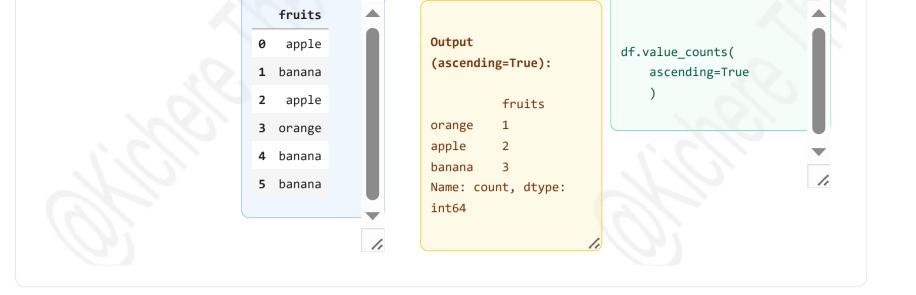




ascending

The **"ascending"** parameter accepts a **Boolean** value and defaults to **False**. It controls the order in which these counts are sorted.

value_counts(ascending=False vs True) Behavior df: When Code (ascending=False): Output fruits ascending=False (ascending=False): apple (default): 1 banana fruits df.value_counts(It sorts from ascending=False banana apple largest to smallest. apple 3 orange # or simply: orange df.value_counts() 4 banana Name: count, dtype: When int64 5 banana ascending=True: It sorts from smallest to largest. Code (ascending=True): df::





dropna

The "dropna" parameter accepts a **Boolean** value and defaults to **True**. It doesn't count rows that have missing values.

value_counts(dropna=True vs False) Behavior

When dropna=True (Default Behavior):

Rows that contain NA values are not counted.

When dropna=False:

Rows that contain NA values are included in the counts.

df:
A
0 1.0
1 2.0
2 NaN
3 2.0
4 3.0
5 NaN
6 3.0

A

2.0 2

3.0 2

1.0 1

Name: count, dtype: int64

Code (dropna=True):

df.value_counts(
 dropna=True
)
or simply:
df.value_counts()

```
Output (dropna=False):
df::
```

```
A
0 1.0
1 2.0
2 NaN
3 2.0
4 3.0
5 NaN
```

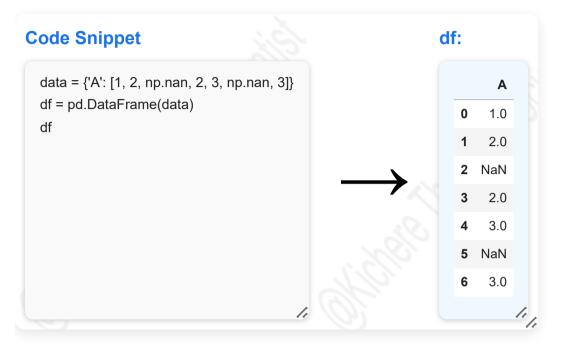
6 3.0

2.0 2
3.0 2
NaN 2
1.0 1
Name: count, dtype: int64

code (dropna=False):

df.value_counts(
 dropna=False
)

//c



Sources & References

```
pandas.DataFrame.value_counts Documentation (
https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.value_counts.html )
```

Contacts and Social Media

Kichere Magubu

- ♠ Dar es Salaam, Tanzania
- in Kichere Magubu
- Kichere The Data Scientist
- kichere-thedatascientist
- kicherethedatascientist@gmail.com
- **** +255 654 729 851

Thank You!

Thank you for reading this e-book!

If you found it valuable,

please consider leaving an honest review.

Your feedback and support mean a lot to me!