None

`pandas.pivot`

The **pandas.pivot()** function reshapes a DataFrame from long to wide format by spreading rows into columns. This is useful for restructuring data for easier analysis or visualization.

Original DataFrame (df_long):

	Name	Subject	Score
0	Fred	Math	85
1	Fred	Science	80
2	Erick	Math	90
3	Erick	Science	85
4	Debra	Math	95
5	Debra	Science	88

$\longrightarrow$

Pivoted DataFrame:

Subject	Math	Science
Name		
Debra	95	88
Erick	90	85
Fred	85	80

Code Snippet:

```
pd.pivot(
    df_long,
    index='Name',
    columns='Subject',
    values='Score'
    )
```

1st Edition

Why This E-book?

"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 self-explanatory. After reviewing them, return to the text to grasp the practical concepts.

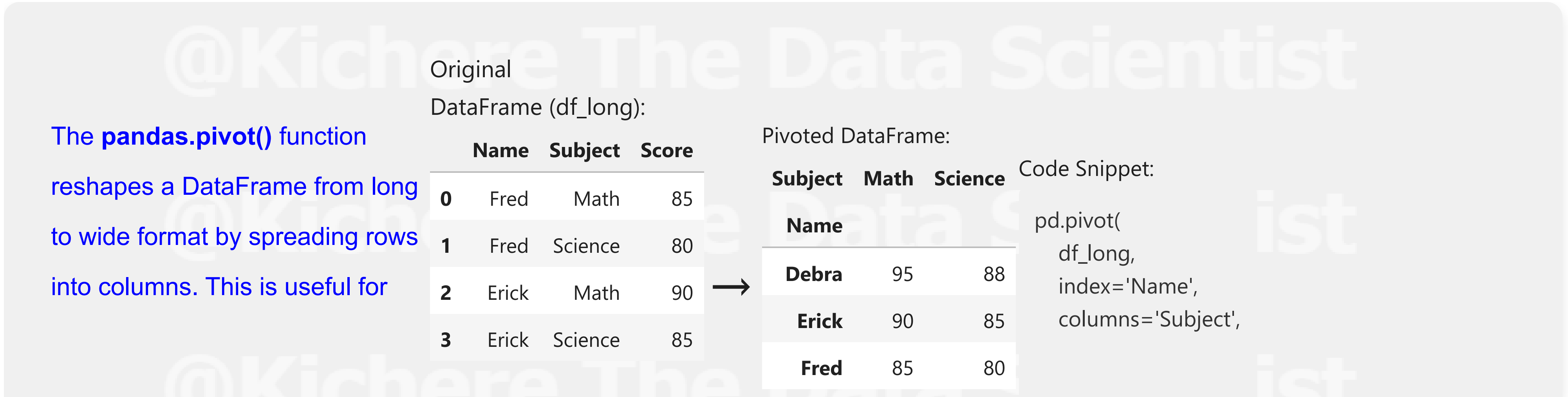
Practice with Datasets: Copy the datasets provided and practice using them to reinforce your understanding.

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

Introduction

```python
#import libraries
import numpy as np
import pandas as pd
```

`pandas.pivot`



The **pandas.pivot()** function reshapes a DataFrame from long to wide format by spreading rows into columns. This is useful for

Original DataFrame (df_long):

	Name	Subject	Score
0	Fred	Math	85
1	Fred	Science	80
2	Erick	Math	90
3	Erick	Science	85

→

Pivoted DataFrame:

Subject	Math	Science
Name		
Debra	95	88
Erick	90	85
Fred	85	80

Code Snippet:

```
pd.pivot(
    df_long,
    index='Name',
    columns='Subject',
```

restructuring data for easier analysis or visualization.

	Name	Subject	Score
4	Debra	Math	95
5	Debra	Science	88

```
values='Score'
)
```

Contents

****

Syntax

```
pandas.pivot(
          data,
        index=None,
        columns=None,
        values=None
        )
```

The data parameter

The DataFrame to be pivoted.

This is the input data in long format, which will be reshaped into a wide format using the pivot function.

The columns parameter

It specifies which columns to use to make new frame’s columns.

It accepts string, object or a list of strings.

When columns is a **string**:

Original DataFrame (df_long):

	Name	Subject	Score
0	Fred	Math	85
1	Fred	Science	80
2	Erick	Math	90
3	Erick	Science	85
4	Debra	Math	95
5	Debra	Science	88

→

Pivoted DataFrame:

Subject	Math	Science
Name		
Debra	95	88
Erick	90	85
Fred	85	80

Code Snippet:

```
pd.pivot(
    df_long,
    index='Name',
    columns='Subject',
    values='Score'
    )
```

Code Snippet:

```
df_long = pd.DataFrame({
    'Name': ['Fred', 'Fred', 'Erick', 'Erick', 'Debra', 'Debra'],
    'Subject': ['Math', 'Science', 'Math', 'Science', 'Math', 'Scienc
e'],
    'Score': [85, 80, 90, 85, 95, 88]
})
df_long
```

→

df_long:

	Name	Subject	Score
0	Fred	Math	85
1	Fred	Science	80
2	Erick	Math	90
3	Erick	Science	85
4	Debra	Math	95
5	Debra	Science	88

When `columns` is a **list of strings**, such as `['Subject', 'Term']`, a multi-level column index is created. Each unique combination of values from these columns becomes part of the new columns in the pivoted DataFrame.

Original DataFrame (df_long):

	Name	Subject	Term	Score
0	Fred	Math	Midterm	85
1	Fred	Science	Midterm	80
2	Erick	Math	Final	90
3	Erick	Science	Final	85
4	Debra	Math	Midterm	95
5	Debra	Science	Midterm	88

$\longrightarrow$

Pivoted DataFrame:

Subject	Math	Science	Math	Science
Term	**Midterm**	**Midterm**	**Final**	**Final**
Name				
Debra	95.0	88.0	NaN	NaN
Erick	NaN	NaN	90.0	85.0
Fred	85.0	80.0	NaN	NaN

Code Snippet:

```
pd.pivot(
    df_long,
    index='Name',
    columns=
    ['Subject', 'Term'],
    values='Score'
    )
```

Code Snippet:

```
df_long = pd.DataFrame({
    'Name': ['Fred', 'Fred', 'Erick', 'Erick', 'Debra', 'Debra'],
    'Subject': ['Math', 'Science', 'Math', 'Science', 'Math', 'Science'],
    'Term': ['Midterm', 'Midterm', 'Final', 'Final', 'Midterm', 'Midterm'],
    'Score': [85, 80, 90, 85, 95, 88]
})
df_long
```

$\longrightarrow$

df_long:

	Name	Subject	Term	Score
0	Fred	Math	Midterm	85
1	Fred	Science	Midterm	80
2	Erick	Math	Final	90
3	Erick	Science	Final	85
4	Debra	Math	Midterm	95
5	Debra	Science	Midterm	88

The `index` parameter

It specifies which column(s) to use to make new frame's index.
If not given, uses existing index.

It accepts string, object or a list of strings and is optional.

When index is a **string**:

Original
DataFrame (df_long):

	Name	Subject	Score
0	Fred	Math	85
1	Fred	Science	80
2	Erick	Math	90
3	Erick	Science	85
4	Debra	Math	95
5	Debra	Science	88

→

Pivoted DataFrame:

Subject	**Math**	**Science**
Name		
Debra	95	88
Erick	90	85
Fred	85	80

Code Snippet:

```
pd.pivot(
   df_long,
   index='Name',
   columns='Subject',
   values='Score'
   )
```

Code Snippet:

```
df_long = pd.DataFrame({
   'Name': ['Fred', 'Fred', 'Erick', 'Erick', 'Debra', 'Debra'],
   'Subject': ['Math', 'Science', 'Math', 'Science', 'Math', 'Scienc
e'],
   'Score': [85, 80, 90, 85, 95, 88]
```

→

df:

	Name	Subject	Score
0	Fred	Math	85
1	Fred	Science	80
2	Erick	Math	90
3	Erick	Science	85

```
})
df_long
```

	Name	Subject	Score
4	Debra	Math	95
5	Debra	Science	88

When `index` is a **list of strings**, such as `['Name', 'Class']`, a multi-level index is created.
This allows you to group by multiple columns, creating a hierarchical structure in the pivoted DataFrame.

Original DataFrame (df_long):

	Name	Class	Subject	Score
0	Fred	10A	Math	85
1	Fred	10A	Science	80
2	Erick	11B	Math	90
3	Erick	11B	Science	85
4	Debra	12C	Math	95
5	Debra	12C	Science	88

→

Pivoted DataFrame:

	Subject	**Math**	**Science**
Name	**Class**		
Debra	**12C**	95	88
Erick	**11B**	90	85
Fred	**10A**	85	80

Code Snippet:

```
pd.pivot(
    df_long,
    index=['Name', 'Class'],
    columns='Subject',
    values='Score'
    )
```

Code Snippet:

```
df_long = pd.DataFrame({
    'Name': ['Fred', 'Fred', 'Erick', 'Erick', 'Debra', 'Debra'],
    'Class': ['10A', '10A', '11B', '11B', '12C', '12C'],
    'Subject': ['Math', 'Science', 'Math', 'Science', 'Math', 'Scienc
e'],
```

→

df_long:

	Name	Class	Subject	Score
0	Fred	10A	Math	85
1	Fred	10A	Science	80
2	Erick	11B	Math	90

```
    'Score': [85, 80, 90, 85, 95, 88]
})
df_long
```

	Name	Class	Subject	Score
3	Erick	11B	Science	85
4	Debra	12C	Math	95
5	Debra	12C	Science	88

When `id_vars` is a tuple:
Here, ("Name", "Year") is a tuple used as identifiers. Both columns remain constant as the other columns are melted. Each row in the melted DataFrame will include the "Name", "Year", the subject, and the corresponding score.

Original DataFrame(df):

Name	Year	Math	Science
Fred	2021	85	80
Erick	2021	90	85
Debra	2022	95	88

$\longrightarrow$

Melted DataFrame:

Name	Year	Subject	Score
Fred	2021	Math	85
Erick	2021	Math	90
Debra	2022	Math	95
Fred	2021	Science	80
Erick	2021	Science	85
Debra	2022	Science	88

Code Snippet:

```
pd.melt(
    df,
    id_vars=('Name', 'Year'),
    var_name='Subject',
    value_name='Score'
    )
```

Code Snippet:

```
df = pd.DataFrame({
    'Name': ['Fred', 'Erick', 'Debra'],
    'Year': [2021, 2021, 2022],
    'Math': [85, 90, 95],
    'Science': [80, 85, 88]
```

$\longrightarrow$

df:

	Name	Year	Math	Science
0	Fred	2021	85	80
1	Erick	2021	90	85

```
})
df
```

	Name	Year	Math	Science
2	Debra	2022	95	88

The values parameter

It specifies which column(s) to use for populating new frame's values.
If not specified, all remaining columns will be used and the result will have hierarchically indexed columns.

It accepts string, object or a list of (strings/objects) and is optional.

Using values as a **string**(one column):

Original
DataFrame (df_long):

	Name	Subject	Score
0	Fred	Math	85
1	Fred	Science	80
2	Erick	Math	90
3	Erick	Science	85
4	Debra	Math	95
5	Debra	Science	88

⟶

Pivoted DataFrame:

Subject	Math	Science
Name		
Debra	95	88
Erick	90	85
Fred	85	80

Code Snippet:

```
pd.pivot(
    df_long,
    index='Name',
    columns='Subject',
    values='Score'
    )
```

df_long:

Code Snippet:

```
df_long = pd.DataFrame({
  'Name': ['Fred', 'Fred', 'Erick', 'Erick', 'Debra', 'Debra'],
  'Subject': ['Math', 'Science', 'Math', 'Science', 'Math', 'Science'],
  'Score': [85, 80, 90, 85, 95, 88]
})
df_long
```

→

	Name	Subject	Score
0	Fred	Math	85
1	Fred	Science	80
2	Erick	Math	90
3	Erick	Science	85
4	Debra	Math	95
5	Debra	Science	88

Using `values` **as a list of strings**

Original DataFrame (df_long):

	Name	Year	Math	Science
0	Fred	2021	85	80
1	Erick	2021	90	85
2	Debra	2022	95	88

→

Pivoted DataFrame:

	Math		**Science**	
Year	**2021**	**2022**	**2021**	**2022**
Name				
Debra	NaN	95.0	NaN	88.0
Erick	90.0	NaN	85.0	NaN
Fred	85.0	NaN	80.0	NaN

Code Snippet:

```
pd.pivot(
    df,
    index='Name',
    columns='Year',
    values=['Math', 'Science']
    )
```

Code Snippet:

```
df = pd.DataFrame({
    'Name': ['Fred', 'Erick', 'Debra'],
```

df:

```
    'Year': [2021, 2021, 2022],
    'Math': [85, 90, 95],
    'Science': [80, 85, 88]
})
df
```

→

	Name	Year	Math	Science
0	Fred	2021	85	80
1	Erick	2021	90	85
2	Debra	2022	95	88

Not specifying `values` (all remaining columns used):

Original DataFrame (df_long):

	Name	Year	Math	Science
0	Fred	2021	85	80
1	Erick	2021	90	85
2	Debra	2022	95	88

→

Pivoted DataFrame:

	Math		**Science**	
Year	**2021**	**2022**	**2021**	**2022**
Name				
Debra	NaN	95.0	NaN	88.0
Erick	90.0	NaN	85.0	NaN
Fred	85.0	NaN	80.0	NaN

Code Snippet:

```
pd.pivot(
    df,
    index='Name',
    columns='Year'
    )
```

Sources & References

pandas.pivot Documentation
(https://pandas.pydata.org/docs/reference/api/pandas.pivot.html)

Contacts and Social Media

Kichere Magubu

Dar es salaam, Tanzania

Kichere Magubu

Kichere The Data Scientist

KichereTheDataScientist

KichereTheDataScientist

kicherethedatascientist@gmail.com

kicherethedatascientist

+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!