None

Contents

Preface

****

Contents(continued)

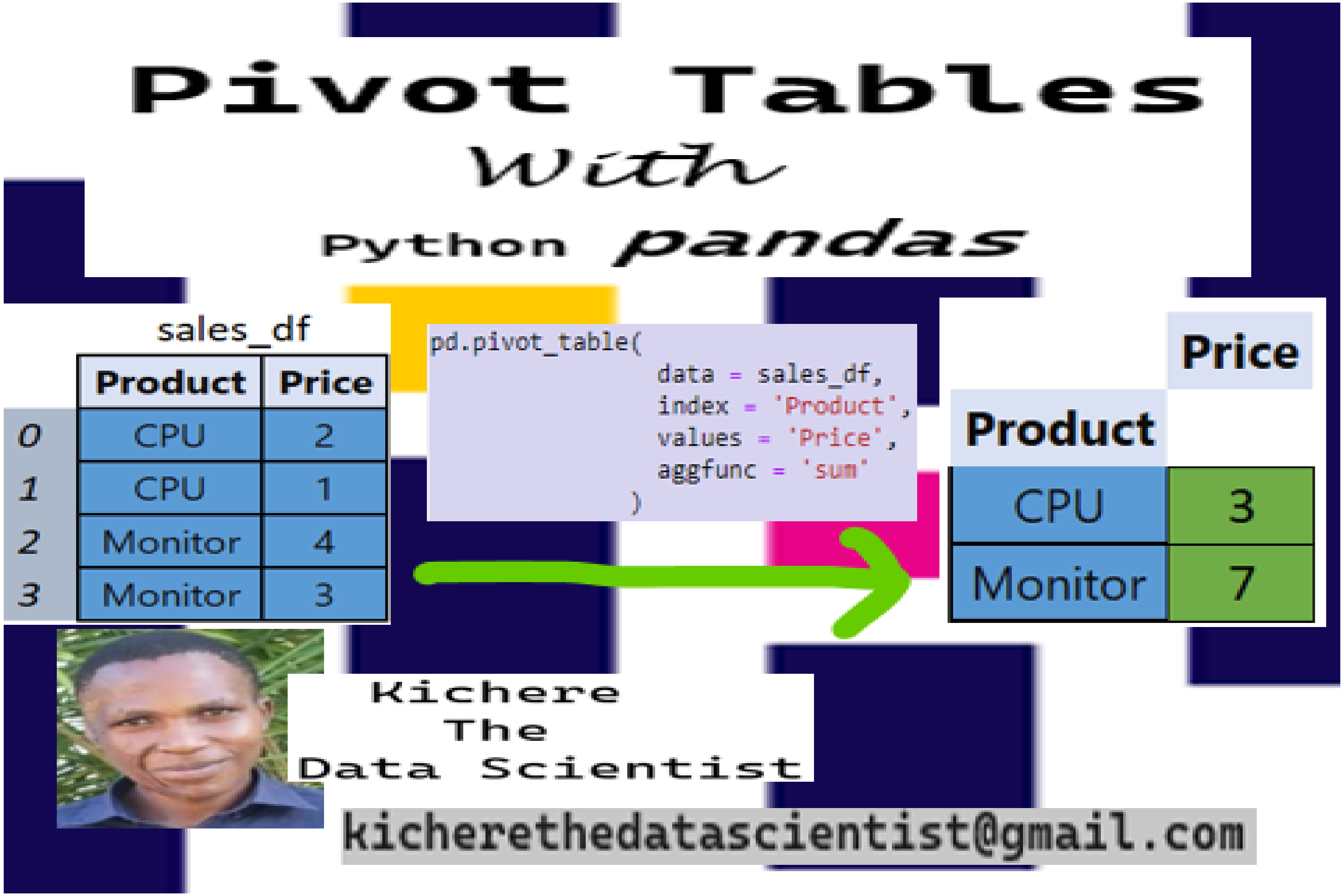
****

Contents(continued)

```
          *************************************************************************************************************************************
```

Introduction

```
#Import libraries
import pandas as pd, numpy as np
from IPython.display import HTML, display
```



	Product	Price
0	CPU	2
1	CPU	1
2	Monitor	4
3	Monitor	3

```
pd.pivot_table(
                data = sales_df,
                index = 'Product',
                values = 'Price',
                aggfunc = 'sum'
              )
```

	Price
Product	
CPU	3
Monitor	7

****

Mastering Pandas pivot tables offers a significant advantage over other tools in data analysis. Unlike other tools like Microsoft Excel,Google Sheets,Tableau, ,Power BI and QlikView/Qlik Sense and Looker, Pandas provides unparalleled flexibility and efficiency when handling large datasets and complex data transformations. Its robust capabilities allow for seamless multi-indexing, advanced aggregations, and customized data summarization, empowering leaders and analysts to derive deeper insights swiftly and effectively.

Compared to alternatives, such as Excel or SQL, Pandas pivot tables excel in scalability and integration within Python's expansive data science ecosystem. This integration not only enhances workflow efficiency but also supports seamless transition between data manipulation, statistical analysis, and machine learning tasks, making it an indispensable tool for leaders striving to achieve data-driven decision-making.

To pivot data is to reorganize and summarize it in various ways, transforming detailed information into a more structured format.

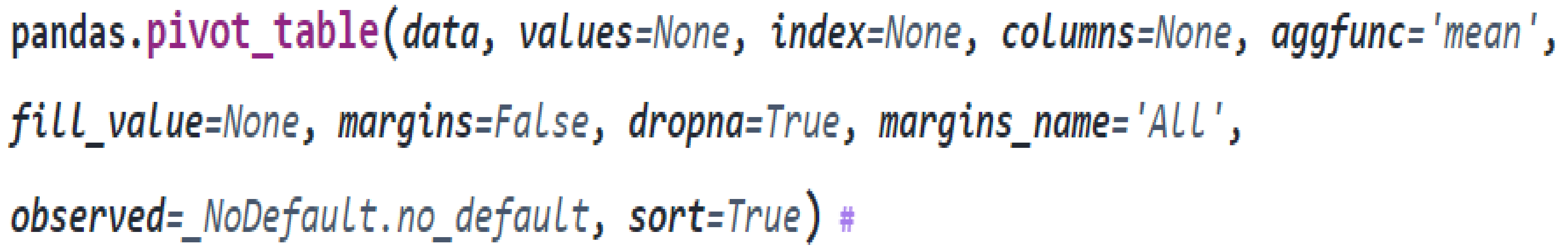
	Category	Sales
0	A	100
1	A	150
2	B	200
3	B	250
4	C	300
5	C	350

⟶

Total Sales for each category:

	Sales
Category	
A	250
B	450
C	650

The "pivot_table" syntax



```
pandas.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean',
fill_value=None, margins=False, dropna=True, margins_name='All',
observed=_NoDefault.no_default, sort=True) #
```

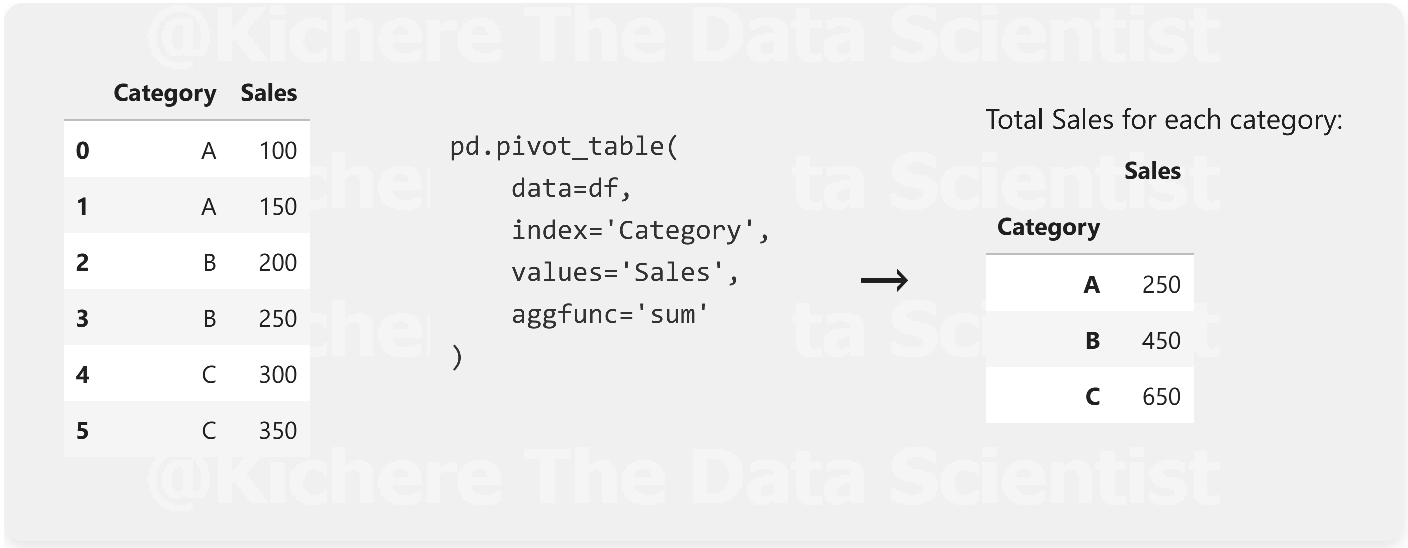
Parameters

```
"""
Parameters:
data
values
index
columns
aggfunc
fill_value
margins
dropna
margins_name
observed
sort
"""
```

The `data` parameter

The "data" parameter accepts a DataFrame
Here "data" parameter is "sales_df"

Compute the total revenue generated from each product in the dataset (sales_df)



	Category	Sales
0	A	100
1	A	150
2	B	200
3	B	250
4	C	300
5	C	350

```
pd.pivot_table(
    data=df,
    index='Category',
    values='Sales',
    aggfunc='sum'
)
```

$\longrightarrow$

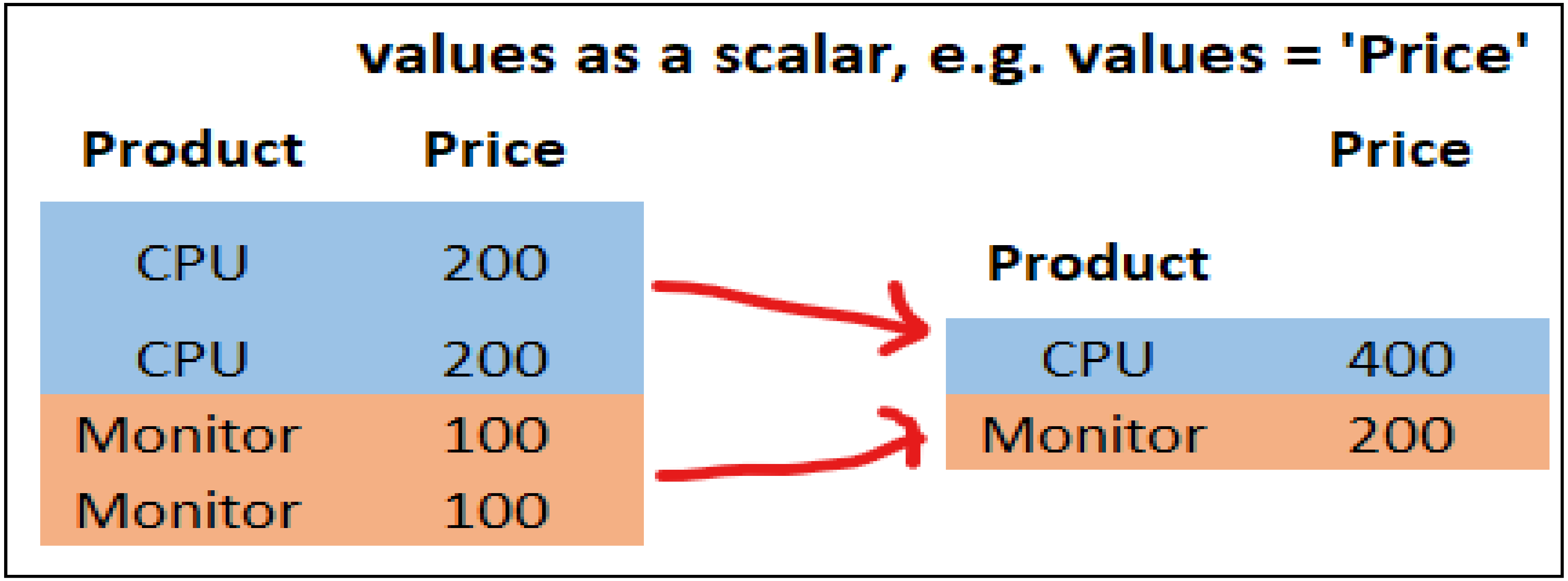
Total Sales for each category:

	Sales
Category	
A	250
B	450
C	650

The values parameter

The "values" parameter accepts scalar values or list-like and is optional.

"values" as a scalar



Can you summarize the total sales value of each product in the dataset 'sales_df'?

values as a scalar:

A scalar is a single, indivisible value, such as a number or a string

sales_df

	Product	Price
0	CPU	200
1	CPU	200
2	Monitor	100
3	Monitor	100

⟶

Total sales summary:

	Price
Product	
CPU	400
Monitor	200

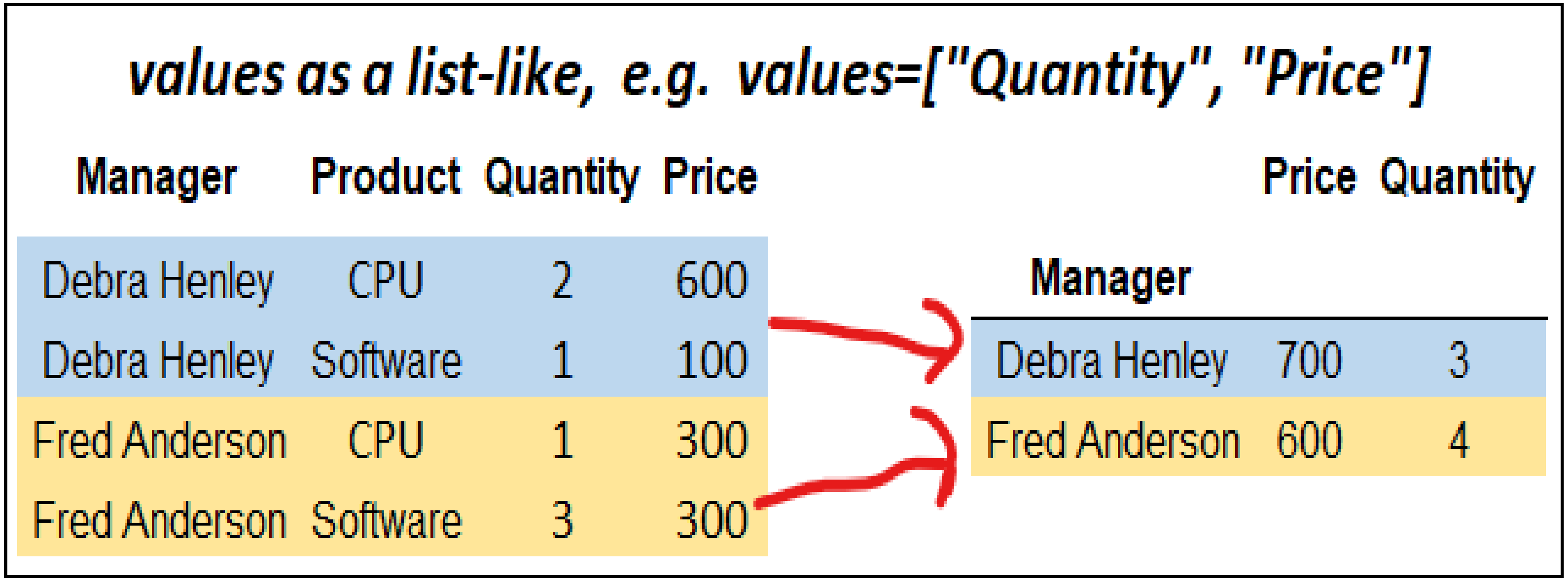
Code Snippet:

```
pd.pivot_table(
    data=sales_df,
    index='Product',
    values='Price',
    aggfunc='sum'
)
```

"values" as a list-like

```
"values" as a list-like e.g

List: [1, 2, 3] or List: ["Price", "Quantity"] for our example

Tuple: (1, 2, 3) or ("Price", "Quantity")

Set: {1, 2, 3}

String: "abc"

Dictionary: {'a': 1, 'b': 2}

NumPy Array: np.array([1, 2, 3])

Pandas Series: pd.Series([1, 2, 3])

Range: range(1, 4)

values=["Price", "Quantity"]
```

Display the total Quantity and Price for each Manager



values as a list-like, e.g. values=["Quantity", "Price"]

Manager	Product	Quantity	Price
Debra Henley	CPU	2	600
Debra Henley	Software	1	100
Fred Anderson	CPU	1	300
Fred Anderson	Software	3	300

	Price	Quantity
Manager		
Debra Henley	700	3
Fred Anderson	600	4

Display the total Quantity and Price for each Manager

sales_df

	Manager	Product	Quantity	Price
0	Debra Henley	CPU	2	600
1	Debra Henley	Software	1	100
2	Fred Anderson	CPU	1	300
3	Fred Anderson	Software	3	300

$\longrightarrow$

Summary:

	Price	Quantity
Manager		
Debra Henley	700	3
Fred Anderson	600	4

Code Snippet:

```
pd.pivot_table(
    data=df,
    index="Manager",
    values=["Quantity", "Price"],
    aggfunc='sum'
)
```

```
#If you omit the values parameter, all columns that are not used as

#indexes or columns in the pivot table will be aggregated:
display(df)
pd.pivot_table(data = df,
               index= 'Product',
               #values = 'Price',
               aggfunc = 'sum'
              )
```

	Manager	Product	Quantity	Price
0	Debra Henley	CPU	2	600
1	Debra Henley	Software	1	100
2	Fred Anderson	CPU	1	300
3	Fred Anderson	Software	3	300

	Manager	Price	Quantity
Product			
CPU	Debra HenleyFred Anderson	900	3
Software	Debra HenleyFred Anderson	400	4

The index parameter

The "index" parameter accepts column name, Grouper, array, or lists of (column names, Groupers and arrays)

Compute the total sales revenue (Price) for each combination of manager and product in the DataFrame 'sales_df'?

Pivot Table:

sales_df

The index parameter:

Specify how data should be grouped

	Manager	Product	Quantity	Price
0	Debra Henley	CPU	2	600
1	Debra Henley	Software	1	100
2	Fred Anderson	CPU	1	300
3	Fred Anderson	Software	3	300

$\longrightarrow$

		Price
Manager	**Product**	
Debra Henley	**CPU**	600
	Software	100
Fred Anderson	**CPU**	300
	Software	300

Therefore

	Manager	Product	Quantity	Price
0	Fred	RAM	2	600
1	Fred	CPU	1	100
2	Debra	CPU	3	900
3	Debra	RAM	3	300

$\longrightarrow$

Summary:

		Price
Manager	**Product**	
Debra	**CPU**	900
	RAM	300
Fred	**CPU**	100
	RAM	600

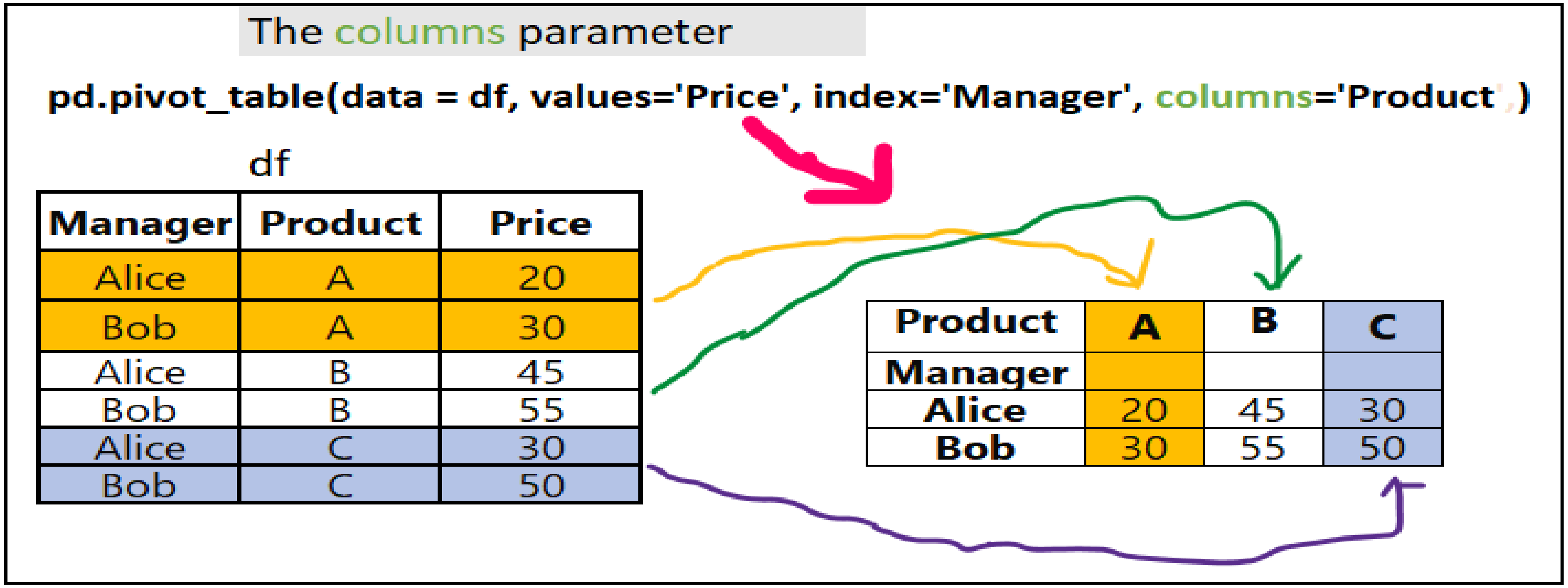
Code snippet:

```
pd.pivot_table(
    sales_df,
    index=['Manager', 'Product'],
    values='Price',
    aggfunc={"Price": sum, "Product": len}
    )
```

The columns parameter

The "columns" parameter accepts column name, Grouper, array, or lists of (column names, Groupers and arrays)

Create a pivot table to show the total Price for each Manager by Product in the DataFrame df?



The columns parameter

pd.pivot_table(data = df, values='Price', index='Manager', columns='Product')

df

Manager	Product	Price
Alice	A	20
Bob	A	30
Alice	B	45
Bob	B	55
Alice	C	30
Bob	C	50

Product	A	B	C
Manager			
Alice	20	45	30
Bob	30	55	50

```python
# Sample data
data = {
    'Manager': ['Alice', 'Bob', 'Alice', 'Bob', 'Alice', 'Bob'],
    'Product': ['A', 'A', 'B', 'B', 'C', 'C'],
    'Price': [20, 30, 45, 55, 30, 50]
}

df = pd.DataFrame(data)
df
```

	Manager	Product	Price
0	Alice	A	20
1	Bob	A	30
2	Alice	B	45
3	Bob	B	55
4	Alice	C	30

	Manager	Product	Price
5	Bob	C	50

```
pd.pivot_table(data=df, values='Price', index='Manager', columns='Product')
```

Product	A	B	C
Manager			
Alice	20.0	45.0	30.0
Bob	30.0	55.0	50.0

```
data = {
    'Manager': ['Debra Henley', 'Debra Henley', 'Debra Henley', 'Debra Henley', 'Debra Henley',
                'Debra Henley', 'Debra Henley', 'Debra Henley', 'Debra Henley', 'Fred Anderson',
                'Fred Anderson'],
    'Product': ['CPU', 'Software', 'Maintenance', 'CPU', 'CPU', 'CPU', 'Software', 'Maintenance',
                'CPU', 'CPU', 'CPU'],
    'Price': [30000, 10000, 5000, 35000, 65000, 40000, 10000, 5000, 35000, 65000, 30000],
    'Status': ['presented', 'presented', 'pending', 'declined', 'won', 'pending', 'presented',
               'pending', 'declined', 'won', 'presented']
}

df = pd.DataFrame(data);df
```

	Manager	Product	Price	Status
0	Debra Henley	CPU	30000	presented
1	Debra Henley	Software	10000	presented
2	Debra Henley	Maintenance	5000	pending
3	Debra Henley	CPU	35000	declined
4	Debra Henley	CPU	65000	won
5	Debra Henley	CPU	40000	pending
6	Debra Henley	Software	10000	presented
7	Debra Henley	Maintenance	5000	pending

	Manager	Product	Price	Status
8	Debra Henley	CPU	35000	declined
9	Fred Anderson	CPU	65000	won
10	Fred Anderson	CPU	30000	presented

```
pd.pivot_table(df, index='Manager', values='Price', columns=['Product', 'Status'], aggfunc='sum')
```

Product				**CPU**	**Maintenance**	**Software**
Status	**declined**	**pending**	**presented**	**won**	**pending**	**presented**
Manager						
Debra Henley	70000.0	40000.0	30000.0	65000.0	10000.0	20000.0
Fred Anderson	NaN	NaN	30000.0	65000.0	NaN	NaN

The aggfunc parameter

You can pass a function, a list of functions, a dictionary, or use the default "mean"

Compute average sales price per manager

When the aggfunc not provided, it calculates the "mean" by default.

sales_df

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	RAM	1	100
2	Fred	CPU	1	300
3	Fred	RAM	3	300

$\longrightarrow$

Pivot Table:

	Price
Manager	
Debra	350.0
Fred	300.0

Code Snippet:

```
pd.pivot_table(
    data=sales_df,
    index="Manager",
    values='Price'
    )
```

Can you summarizes total sales by manager?

When you pass a function e.g "sum"

sales_df:

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	RAM	1	100
2	Fred	CPU	1	300
3	Fred	RAM	3	300

$\longrightarrow$

Pivot Table:

	Price
Manager	
Debra	700
Fred	600

Code Snippet:

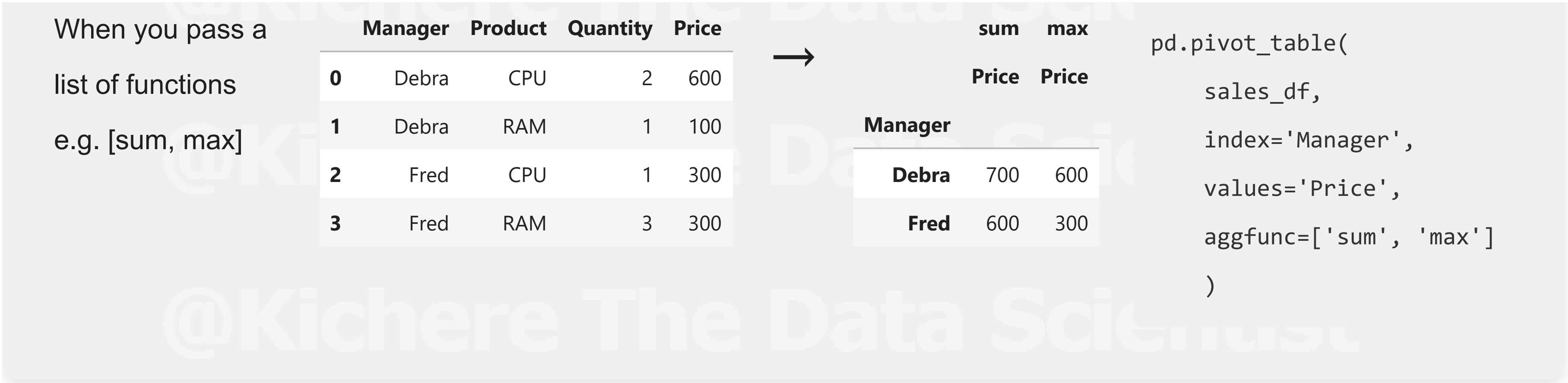
```
pd.pivot_table(
    sales_df,
    index='Manager',
    values='Price',
    aggfunc= 'sum'
    )
```

Compute the highest and total values of 'Price' for each 'Manager'?

sales_df:

Pivot Table:

Code Snippet:



When you pass a list of functions e.g. [sum, max]

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	RAM	1	100
2	Fred	CPU	1	300
3	Fred	RAM	3	300

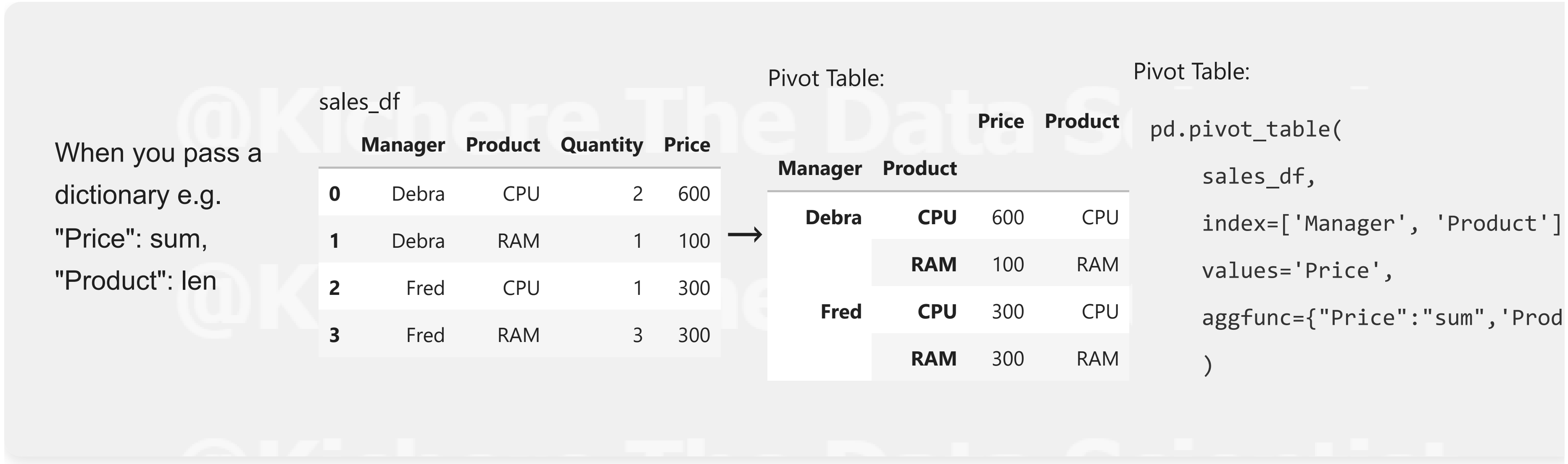
$\longrightarrow$

	sum	max
	Price	**Price**
Manager		
Debra	700	600
Fred	600	300

```
pd.pivot_table(
    sales_df,
    index='Manager',
    values='Price',
    aggfunc=['sum', 'max']
)
```

```
You could also find 'sum', 'max', 'min', 'mean', 'median', 'count', 'std', 'var'
```

calculates the total sales price and maximum product grouped by manager and product



When you pass a dictionary e.g. "Price": sum, "Product": len

sales_df

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	RAM	1	100
2	Fred	CPU	1	300
3	Fred	RAM	3	300

$\longrightarrow$

Pivot Table:

		Price	Product
Manager	**Product**		
Debra	**CPU**	600	CPU
	RAM	100	RAM
Fred	**CPU**	300	CPU
	RAM	300	RAM

Pivot Table:

```
pd.pivot_table(
    sales_df,
    index=['Manager', 'Product']
    values='Price',
    aggfunc={"Price":"sum",'Prod
)
```

```
You could also find 'sum', 'max', 'min', 'mean', 'median', 'count', 'std', 'var'
```

The fill_value parameter

The "fill_value" parameter specifies the value to replace missing values (NaN) in the resulting pivot table
It accepts a scalar value (single value), if no other value is provided, None is used as the default value.

*

Summarize the total 'Quantity' of products, grouped by 'Manager' across different 'Product' and 'Price' combinations

Dou you see the (NANs) in a resulting pivot table?

sales_df:

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	RAM	1	100
2	Fred	CPU	1	300
3	Fred	RAM	3	300

$\longrightarrow$

Pivot Table:

Product	**CPU**		**RAM**	
Price	**300**	**600**	**100**	**300**
Manager				
Debra	NaN	2.0	1.0	NaN
Fred	1.0	NaN	NaN	3.0

**

Summarize the total 'Quantity' of products, grouped by 'Manager' across different 'Product' and 'Price' combinations

Replace the (NANs) with a double dash (--)

sales_df:

Product	**CPU**		**RAM**	
Price	**300**	**600**	**100**	**300**
Manager				
Debra	NaN	2.0	1.0	NaN
Fred	1.0	NaN	NaN	3.0

Code Snippet:

```
pd.pivot_table(
    sales_df,
    index=["Manager"],
    columns=['Product', 'Pric
e'],
    values='Quantity',
    aggfunc='sum',
    fill_value='--'
)
```

$\rightarrow$

Pivot Table:

Product	**CPU**		**RAM**	
Price	**300**	**600**	**100**	**300**
Manager				
Debra	--	2	1	--
Fred	1	--	--	3

The $\boxed{\text{margins}}$ parameter

The "margins" parameter adds totals for each row and column.
It boolean and defaults to False, adds totals for each row and column when set to True.

*

Show the total sales (sum of 'Price') for each manager across different products.

Without margins or when `margins=False`.

sales_df:

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	RAM	1	100
2	Fred	CPU	1	300
3	Fred	RAM	3	300

Code Snippet:

```
pd.pivot_table(
    sales_df,
    index='Manager',
    columns='Product',
    values='Price',
    aggfunc='sum'
)
```

Pivot Table:

Product	CPU	RAM
Manager		
Debra	600	100
Fred	300	300

*

Show the total sales (sum of 'Price') for each manager across different products

If `margins=True,` special `All` columns and rows will be added with partial group aggregates across the categories on the rows and columns.

sales_df:

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	RAM	1	100
2	Fred	CPU	1	300
3	Fred	RAM	3	300

Code Snippet:

```
pd.pivot_table(
    sales_df,
    index='Manager',
    columns='Product',
    values='Price',
    aggfunc='sum',
    margins=True
)
```

Pivot Table:

Product	CPU	RAM	All
Manager			
Debra	600	100	700
Fred	300	300	600
All	900	400	1300

The `dropna` parameter

The "dropna" parameter does not include columns whose entries are all NaN. If True, rows with a NaN value in any column will be omitted before computing margins.
It is a boolean option and defaults to True.

*

Show the total sales (sum of 'Price') for each manager across different products.

If `margins=True` (default), Pandas excludes rows or columns with NaN values before computing a pivot table. This ensures NaNs do not affect calculations,and the resulting table is based only on available data without NaNs.

sales_df:

	Manager	Product	Quantity	Price	Status
0	Debra	CPU	2.0	600.0	None
1	Debra	RAM	NaN	100.0	None
2	Fred	CPU	1.0	300.0	None
3	Fred	RAM	3.0	NaN	None
4	None	None	NaN	NaN	None

Code Snippet:

```
pd.pivot_table(
    sales_df,
    index='Manage
r',
    columns='Produc
t',
    values='Price',
    aggfunc='sum',
    margins=True,
    dropna=True
)
```

Pivot Table:

Product	CPU	RAM	All
Manager			
Debra	600.0	100.0	700.0
Fred	300.0	0.0	300.0
All	900.0	100.0	1000.0

**

With `dropna=False`, NaN values are included in the pivot table

sales_df:

	Manager	Product	Quantity	Price	Status
0	Debra	CPU	2.0	600.0	None
1	Debra	RAM	NaN	100.0	None
2	Fred	CPU	1.0	300.0	None
3	Fred	RAM	3.0	NaN	None
4	None	None	NaN	NaN	None

Code Snippet:

```
pd.pivot_table(
    sales_df,
    index='Manager',
    columns=['Product',
'Status'],
    values='Price',
    aggfunc='sum',
    margins=True,
    dropna=False
)
```

Pivot Table:

Product	CPU	RAM	NaN	All
Status	NaN	NaN	NaN	
Manager				
Debra	600.0	100.0	NaN	700.0
Fred	300.0	0.0	NaN	300.0
NaN	NaN	NaN	0.0	NaN
All	NaN	NaN	NaN	1000.0

The margins_name parameter

The "margins_name" parameter specifies the name of the row or column that will contain the totals when margins=True."
By default, the name is set to 'All'.

*

Show the total sales (sum of 'Price') for each manager across different products

The margins_name parameter:
Allows customization of the label used for the
totals row or column when margins=True.
This parameter is useful for providing clear and descriptive labels in pivot table summaries.

sales_df:

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	RAM	1	100
2	Fred	CPU	1	300
3	Fred	RAM	3	300

Code Snippet:

```
pd.pivot_table(
    sales_df,
    index='Manager',
    columns='Product',
    values='Price',
    aggfunc='sum',
    margins=True,
    margins_name='TotaL'
)
```

Pivot Table:

Product	CP
Manager	
Debra	60
Fred	30
TotaL	90

The sort parameter

The "sort" parameter specifies if the result should be sorted
By default, "sort" is set to True

Price of products sold, grouped by manager and product?

sales_df: **Code Snippet:** **Pivot Table:**

Setting (sort=True) (default), ensures that the data is presented in an organized manner based on the index labels 'Manager' and 'Product'.

	Manager	Product	Quantity	Price
0	Fred	RAM	2	600
1	Fred	CPU	1	100
2	Debra	CPU	3	900
3	Debra	RAM	3	300

```
pd.pivot_table(
    sales_df,
    values='Price',
    index=['Manager', 'Product'],
    aggfunc='sum',
    sort=True
)
```

		Price
Manager	**Product**	
Debra	**CPU**	900
	RAM	300
Fred	**CPU**	100
	RAM	600

Price of products sold, grouped by manager and product?

Setting (sort=False), maintains the original order of the data

sales_df:

	Manager	Product	Quantity	Price
0	Fred	RAM	2	600
1	Fred	CPU	1	100
2	Debra	CPU	3	900
3	Debra	RAM	3	300

Code Snippet:

```
pd.pivot_table(
    sales_df,
    values='Price',
    index=['Manager', 'Product'],
    aggfunc='sum',
    sort=False
)
```

Pivot Table:

		Price
Manager	**Product**	
Debra	**CPU**	600
	RAM	100
Fred	**CPU**	300
	RAM	300

The observed parameter

The "observed" parameter is Deprecated since version 2.2.0: my pandas version = 2.2.2

```
#print(pd.__version__)
#2.2.2
```

```
data = {
    'Region': ['East', 'East', 'West', 'West', 'North'],
    'Salesperson': ['Alice', 'Bob', 'Alice', 'Charlie', 'David'],
    'Sales': [10000, 15000, 12000, 8000, 9000]
}

df = pd.DataFrame(data)
df
```

	Region	Salesperson	Sales
0	East	Alice	10000
1	East	Bob	15000
2	West	Alice	12000
3	West	Charlie	8000
4	North	David	9000

```
# Pivot table with observed=True
df.pivot_table(index='Region', columns='Salesperson',
               values='Sales', aggfunc='sum', observed=True)
```

Salesperson	Alice	Bob	Charlie	David
Region				
East	10000.0	15000.0	NaN	NaN
North	NaN	NaN	NaN	9000.0
West	12000.0	NaN	8000.0	NaN

```
# Pivot table with observed=False
df.pivot_table(index='Region', columns='Salesperson',
               values='Sales', aggfunc='sum', observed=False)
```

Salesperson	Alice	Bob	Charlie	David
Region				
East	10000.0	15000.0	NaN	NaN
North	NaN	NaN	NaN	9000.0
West	12000.0	NaN	8000.0	NaN

```
#Deprecated since version 2.2.0: The default value of False is deprecated and
#will change to True in a future version of pandas.
```

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

Powered by
Eastern Africa Statistical Training Centre
SKT Tanzania Ltd

****

Why This E-book?

> **"The aim of this ebook is to give you the 'aha' moment right away**
> **concept."**

- **Practical step By Step Guide With Simple Examples**
- **Visual Illustrations and Interactive**
- **Simple Datasets**
- **Comprehensive Coverage**
- **Designed for Beginners**
- **No Prior Knowledge Required**
- **Includes Pandas Documentation**

****

Project(Real Life application)

Practical Business Python

Pandas Pivot Table Explained (https://pbpython.com/pandas-pivot-table-explained.html)

Acknowledgments

First and foremost, I express my gratitude to God **for His blessings and guidance throughout this journey. I am also deeply thankful to the** Eastern Africa Statistical Training Centre **and** SKT Tanzania Ltd **for their invaluable support. Special thanks to my** parents **whose unwavering encouragement and support have been the cornerstone of this endeavor.**

Sources & References

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

Pandas Pivot Table Explained (https://pbpython.com/pandas-pivot-table-explained.html)

Author Biography

Kichere Magubu is a data enthusiast and content creator.

Thank you!

Thank you for reading this e-book!.

It is designed to assist you in understanding pivot tables using Python pandas. If you found this book valuable and beneficial, I kindly request that you take a moment to review it. Your honest feedback is highly appreciated and makes a significant difference to me.

Please take a minute to write me an honest review. Your support means the world to me!

Thank you!

```
#print("The cell to convert jupyter notebook to html")
!jupyter nbconvert --to hide_code_html "Pivot Tables.ipynb"
```