None

Contents

****

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

Why This E-book?

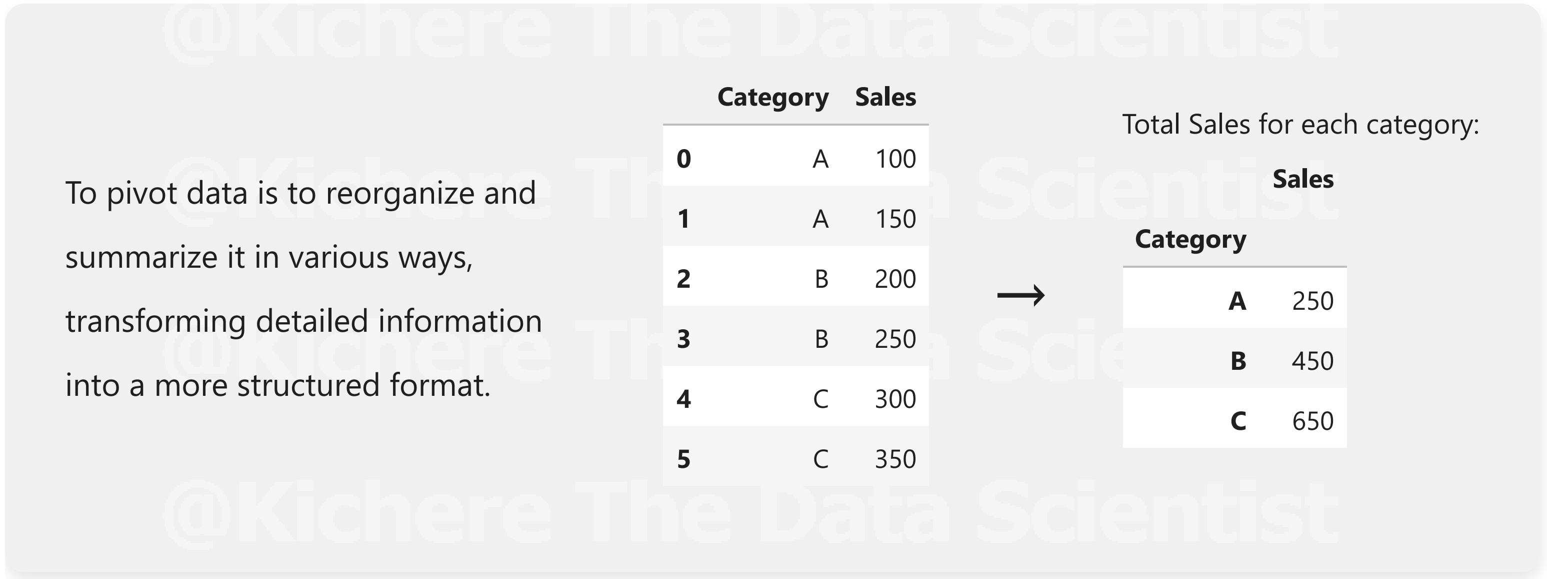
"The aim of this ebook is to give you the 'aha' moment right away at the start of learning a new concept."

- Practical step By Step Guide With Simple Examples
- Visual Illustrations and Interactive
- Simple Datasets
- Comprehensive Coverage (pandas Documentation used as reference)

****

Introduction

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



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

$\longrightarrow$

Total Sales for each category:

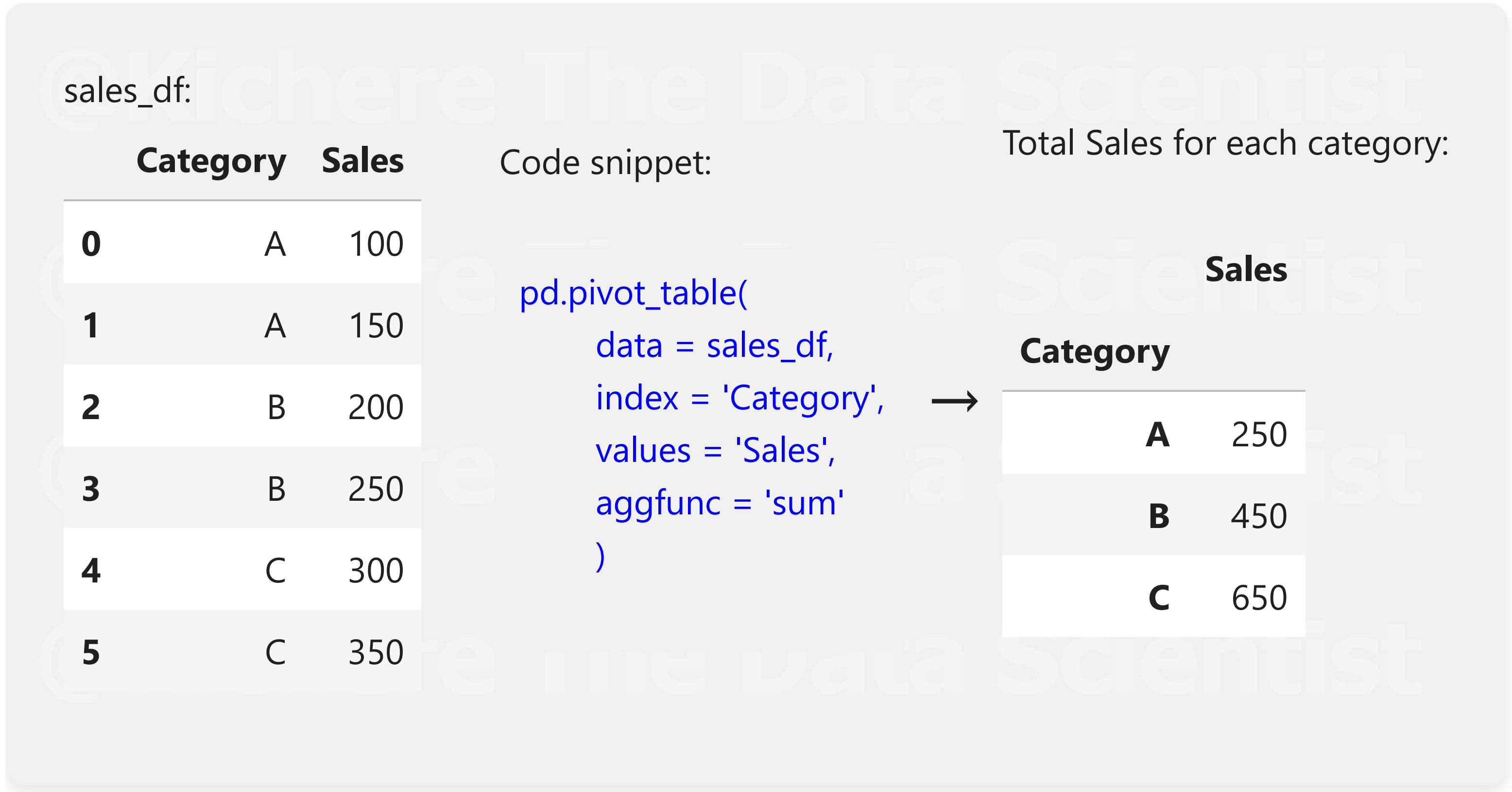
Category	Sales
A	250
B	450
C	650

The "pivot_table" syntax

```python
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) #
```

The data parameter

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



sales_df:

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

Code snippet:

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

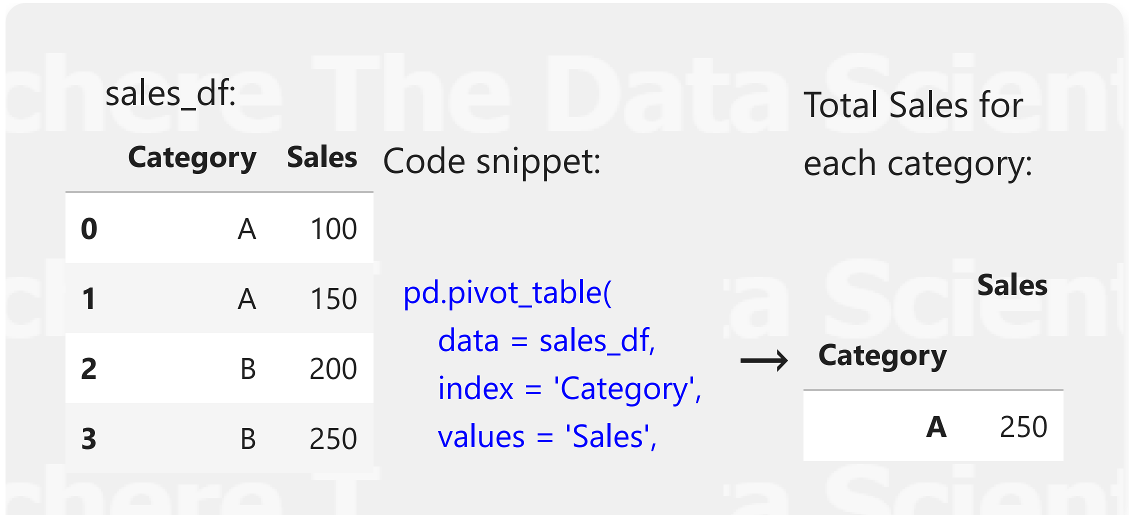
$\longrightarrow$

Total Sales for each category:

	Sales
Category	
A	250
B	450
C	650

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

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



sales_df:

	Category	Sales
0	A	100
1	A	150
2	B	200
3	B	250

Code snippet:

```
pd.pivot_table(
    data = sales_df,
    index = 'Category',
    values = 'Sales',
```

$\longrightarrow$

Total Sales for each category:

	Sales
Category	
A	250

	Category	Sales
4	C	300
5	C	350

```
aggfunc = 'sum'
)
```

	Sales
Category	
B	450
C	650

sales_df:

Code Snippet:

```
data = {
    'Category': ['A', 'A', 'B', 'B', 'C', 'C'],
    'Sales': [100, 150, 200, 250, 300, 350]
}
sales_df = pd.DataFrame(data)
sales_df
```

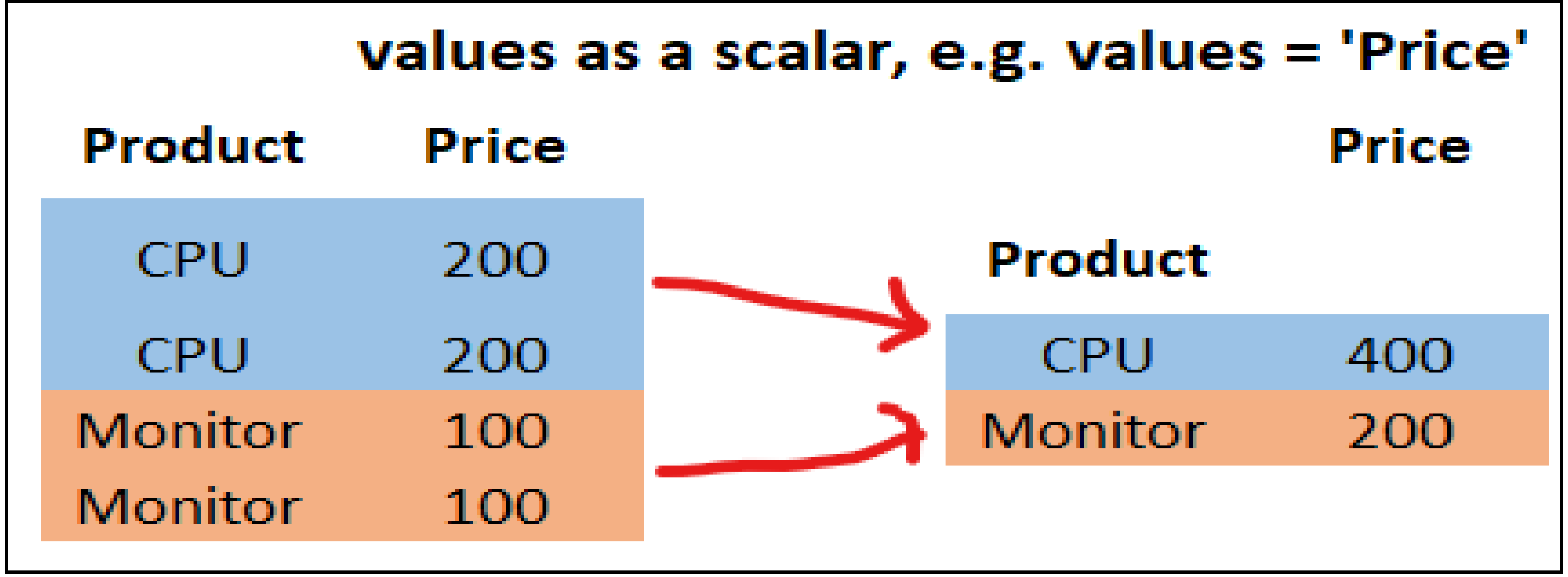
$\longrightarrow$

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

The values parameter

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

"values" as a scalar



values as a scalar, e.g. values = 'Price'

Product	Price
CPU	200
CPU	200
Monitor	100
Monitor	100

	Price
Product	
CPU	400
Monitor	200

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:

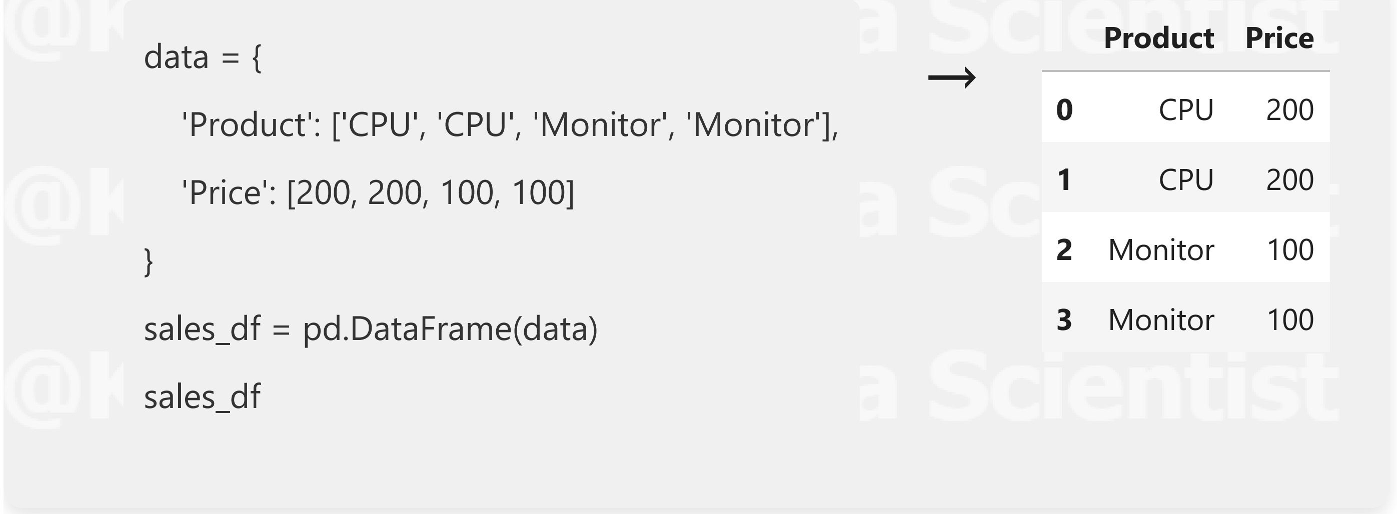
	Price
Product	
CPU	400
Monitor	200

Code Snippet:

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

Code Snippet:

sales_df:



```
data = {
    'Product': ['CPU', 'CPU', 'Monitor', 'Monitor'],
    'Price': [200, 200, 100, 100]
}
sales_df = pd.DataFrame(data)
sales_df
```

→

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

"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}

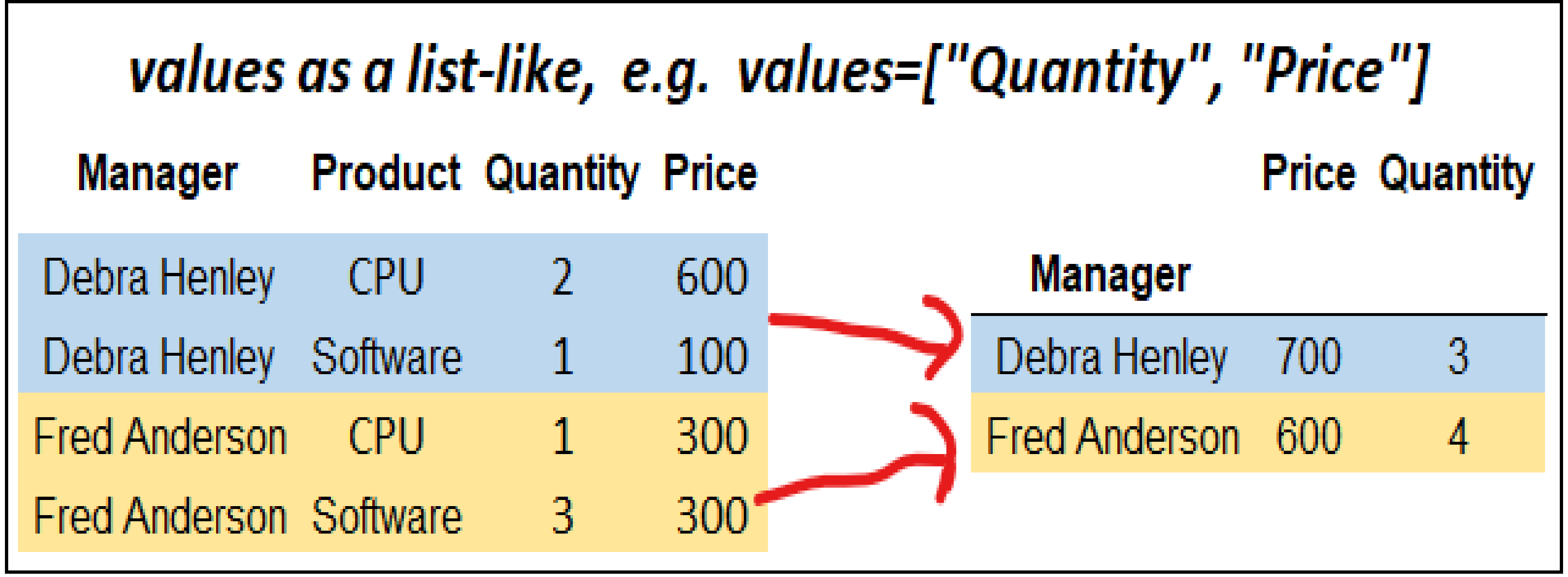
NumPy Array: np.array([1, 2, 3])

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	CPU	2	600
1	Debra	Monitor	1	100
2	Fred	CPU	1	300
3	Fred	Monitor	3	300

$\longrightarrow$

Summary:

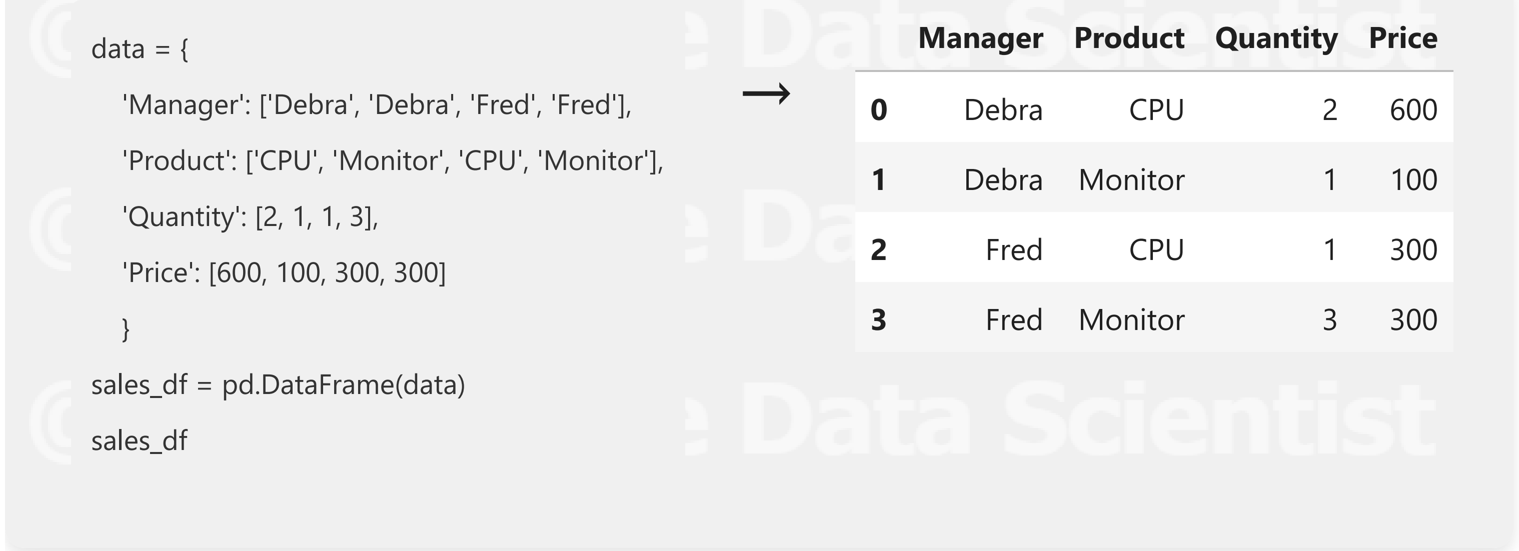
	Price	Quantity
Manager		
Debra	700	3
Fred	600	4

Code Snippet:

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

Code Snippet:

sales_df:



```
data = {
    'Manager': ['Debra', 'Debra', 'Fred', 'Fred'],
    'Product': ['CPU', 'Monitor', 'CPU', 'Monitor'],
    'Quantity': [2, 1, 1, 3],
    'Price': [600, 100, 300, 300]
    }
sales_df = pd.DataFrame(data)
sales_df
```

→

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

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

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

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

	Manager	Price	Quantity
Product			
CPU	DebraFred	900	3
Monitor	DebraFred	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'?

The index parameter:

Specify how data should be grouped

sales_df

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

$\longrightarrow$

Pivot Table:

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

Therefore

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

$\longrightarrow$

Summary:

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

Code snippet:

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

Code Snippet:

sales_df:

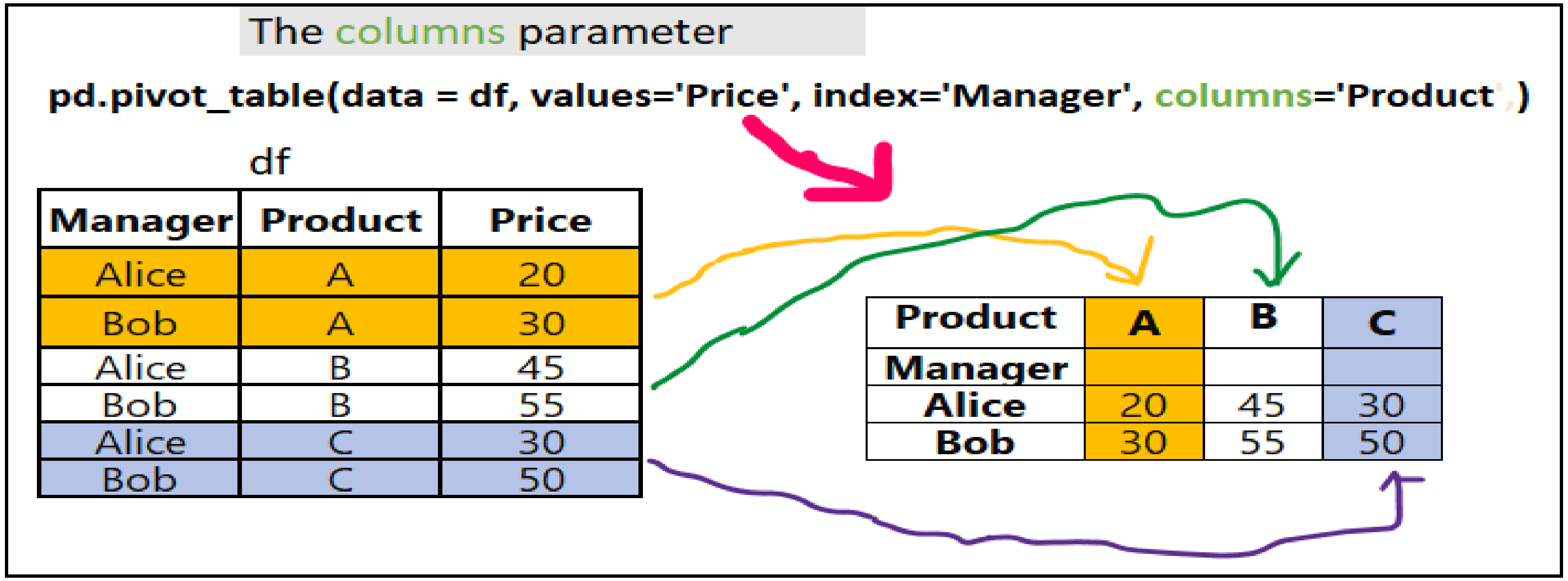
```
data = {
    'Manager': ['Debra', 'Debra', 'Fred', 'Fred'],
    'Product': ['CPU', 'Monitor', 'CPU', 'Monitor'],
    'Quantity': [2, 1, 1, 3],
    'Price': [600, 100, 300, 300]
    }
sales_df = pd.DataFrame(data)
sales_df
```

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

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

```
# 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
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

	Manager	Product	Price	Status
7	Debra Henley	Maintenance	5000	pending
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'
    )
```

Code Snippet:

```
data = {
    'Manager': ['Debra', 'Debra', 'Fred', 'Fred'],
    'Product': ['CPU', 'Monitor', 'CPU', 'Monitor'],
    'Quantity': [2, 1, 1, 3],
    'Price': [600, 100, 300, 300]
    }
sales_df = pd.DataFrame(data)
sales_df
```

$\longrightarrow$

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

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

$\longrightarrow$

Pivot Table:

	Price
Manager	
Debra	700

Code Snippet:

```
pd.pivot_table(
    sales_df,
    index='Manager',
```

	Manager	Product	Quantity	Price
2	Fred	CPU	1	300
3	Fred	RAM	3	300

	Price
Manager	
Fred	600

```
values='Price',
aggfunc= 'sum'
)
```

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

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

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:

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

Code Snippet:

```
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",'Product':"max"}
    )
```

```
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', 'Price'],
    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 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='Manager',
    columns='Product',
    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

Code Snippet:

```
data = {
    'Manager': ['Debra', 'Debra', 'Fred', 'Fred', None],
    'Product': ['CPU', 'RAM', 'CPU', 'RAM', None],
    'Quantity': [2, None, 1, 3, None],
    'Price': [600, 100, 300, None, None],
    'Status': [None, None, None, None, None]
}
sales_df = pd.DataFrame(data)
sales_df
```

→

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

**

With `dropna=False`, NaN values are included in

sales_df:

	Manager	Product	Quantity	Price	Status
0	Debra	CPU	2.0	600.0	None
1	Debra	RAM	NaN	100.0	None

Code Snippet:

```
pd.pivot_table(
    sales_df,
    index='Manager',
    columns=['Product', 'Status'],
    values='Price',
```

Pivot Table:

Product	CPU	RAM	NaN	All
Status	NaN	NaN	NaN	
Manager				
Debra	600.0	100.0	NaN	700.0

the pivot table

	Manager	Product	Quantity	Price	Status
2	Fred	CPU	1.0	300.0	None
3	Fred	RAM	3.0	NaN	None
4	None	None	NaN	NaN	None

```
aggfunc='sum',
margins=True,
dropna=False
)
```

Product	CPU	RAM	NaN	All
Status	**NaN**	**NaN**	**NaN**	
Manager				
Fred	300.0	0.0	NaN	300.0
NaN	NaN	NaN	0.0	NaN
All	NaN	NaN	NaN	1000.0

Code Snippet:

```
data = {
    'Manager': ['Debra', 'Debra', 'Fred', 'Fred', None],
    'Product': ['CPU', 'RAM', 'CPU', 'RAM', None],
    'Quantity': [2, None, 1, 3, None],
    'Price': [600, 100, 300, None, None],
    'Status': [None, None, None, None, None]
}
sales_df = pd.DataFrame(data)
sales_df
```

→

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

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

Code Snippet:

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

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

Pivot Table:

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

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?

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

sales_df:

	Manager	Product	Quantity	Price
0	Fred	CPU	1	100
1	Debra	RAM	3	300
2	Fred	RAM	2	600
3	Debra	CPU	3	900
4	Fred	SSD	5	1200
5	Debra	SSD	7	1300

Code Snippet:

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

Pivot Table:

		Price
Manager	**Product**	
Debra	**CPU**	900
	RAM	300
	SSD	1300
Fred	**CPU**	100
	RAM	600
	SSD	1200

Code Snippet:

```
data = {
    'Manager': ['Debra', 'Debra', 'Fred', 'Fred', None],
    'Product': ['CPU', 'RAM', 'CPU', 'RAM', None],
    'Quantity': [2, None, 1, 3, None],
    'Price': [600, 100, 300, None, None],
    'Status': [None, None, None, None, None]
}
sales_df = pd.DataFrame(data)
sales_df
```

$\longrightarrow$

sales_df:

	Manager	Product	Quantity	Price
0	Fred	CPU	1	100
1	Debra	RAM	3	300
2	Fred	RAM	2	600
3	Debra	CPU	3	900
4	Fred	SSD	5	1200
5	Debra	SSD	7	1300

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:

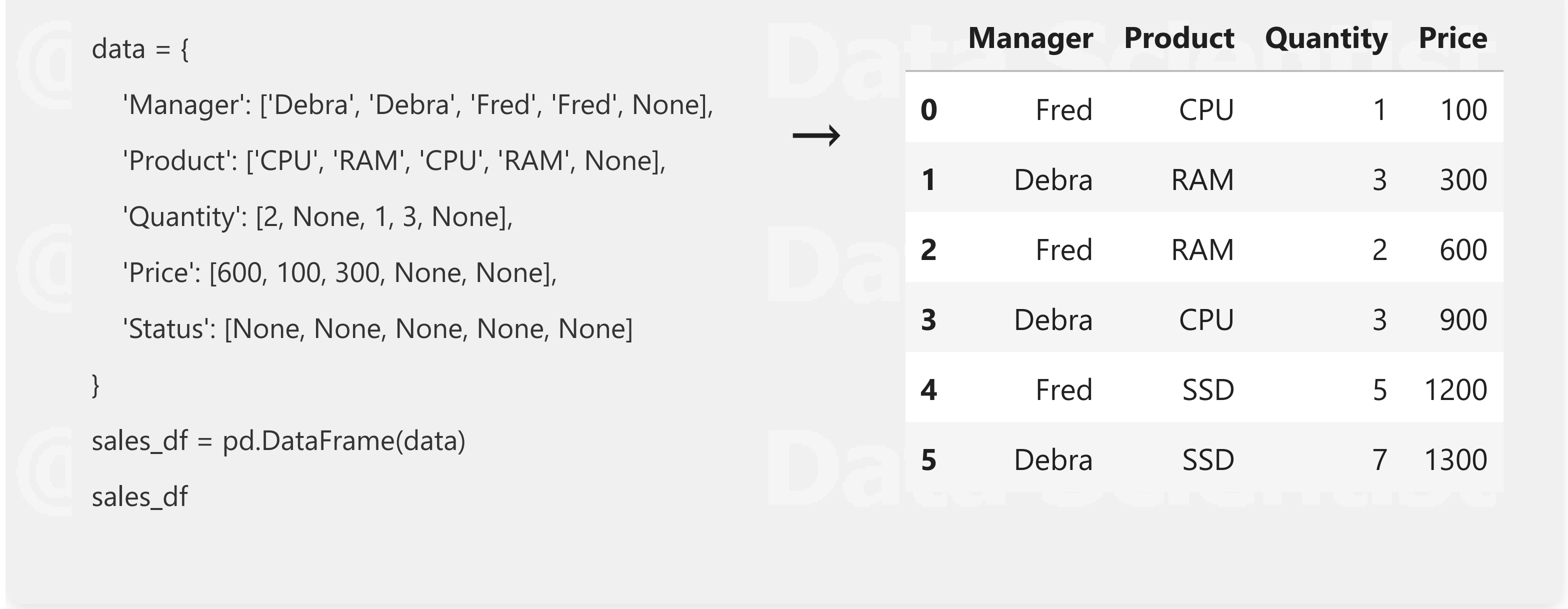
```
pd.pivot_table(
    sales_df,
    values='Price',
    index=['Manager', 'Product'],
    aggfunc='sum',
    sort=False
)
```

Pivot Table:

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

Code Snippet:

sales_df:



```
data = {
    'Manager': ['Debra', 'Debra', 'Fred', 'Fred', None],
    'Product': ['CPU', 'RAM', 'CPU', 'RAM', None],
    'Quantity': [2, None, 1, 3, None],
    'Price': [600, 100, 300, None, None],
    'Status': [None, None, None, None, None]
}
sales_df = pd.DataFrame(data)
sales_df
```

→

	Manager	Product	Quantity	Price
0	Fred	CPU	1	100
1	Debra	RAM	3	300
2	Fred	RAM	2	600
3	Debra	CPU	3	900
4	Fred	SSD	5	1200
5	Debra	SSD	7	1300

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

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

```python
# 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

```python
# 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

```python
#Deprecated since version 2.2.0: The default
#value of False is deprecated and
```

```
#will change to True in a
#future version of pandas.
```

"pandas.pivot_table()" vs "pandas.pivot()"

simple reshaping: When Data has no duplicates

→

df:

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

Simple reshaping:

Product	CPU	Monitor
Manager		
Debra	600	100
Fred	300	300

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

Reshaping data with aggregation:

Product	CPU	Monitor
Manager		
Debra	600	100
Fred	300	300

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

Data with duplicates

df:

Throws an error if duplicates exist:

```
ValueError:
Index contains
```

Handles duplicates using aggregation (e.g., sum, mean):

	Manager	Product	Quantity	Price
0	Debra	CPU	2	600
1	Debra	CPU	1	300
2	Debra	Monitor	1	100
3	Fred	Monitor	3	300
4	Fred	Monitor	1	150

→

duplicate entries,
cannot reshape

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

Product	**CPU**	**Monitor**
Manager		
Debra	900.0	100.0
Fred	NaN	450.0

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

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

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

Project(Real Life application)

Practical Business Python

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

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)

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