Unpivoting and pivoting in

Power Query (M)
Python
R

Unpivoting is a common operation

Change this

	Item	202110	202111	202112	202201	202202	202203
1	A	1	2	4	2	3	4
2	B	2	4	8			
3	C	3	6	12			
4	D	4	8	16			
5	E	5	10	20			
6	A	*NULL*	*NULL*	*NULL*			
7	D	*NULL*	*NULL*	*NULL*			
8	E	*NULL*	*NULL*	*NULL*			

To this

	Item	YearMonth	RawValue
1	A	202110	1
2	A	202111	2
3	A	202112	4
4	A	202201	2
5	A	202202	3
6	A	202203	4
7	B	202110	2
8	B	202111	4
9	B	202112	8
10	B	202201	4

It turns out that building familiarity with one language can make it easier to pick up another.

Let’s compare Power Query (M), Python and R

M

Column name for the unpivoted attribute

The data as a table

```
Table.UnpivotOtherColumns(df, {"Item"}, "YearMonth", "Value")
```

ID variables (not unpivoted and repeated in each stack)

Column name for the values from the unpivoted attribute

Python

The data as a Pandas dataframe

Column name for the unpivoted attribute

```
df.melt("Item", var_name = "YearMonth")
```

ID variables (not unpivoted, repeated in each stack)

R

The data as a dataframe

Column name for the unpivoted attribute

```
melt(df, variable.name = 'YearMonth', id = 'Item')
```

ID variables (not unpivoted, repeated in each stack)

Syntax shown is for data.table package.

reshape2 uses “id.vars” instead of “id”.

Let's do it the other way!

Get this

	Item	202110	202111	202112	202201	202202	202203
1	A	1	2	4	2	3	4
2	B	2	4	8			
3	C	3	6	12			
4	D	4	8	16			
5	E	5	10	20			
6	A	*NULL*	*NULL*	*NULL*			
7	D	*NULL*	*NULL*	*NULL*			
8	E	*NULL*	*NULL*	*NULL*			

From this

	Item	YearMonth	RawValue
1	A	202110	1
2	A	202111	2
3	A	202112	4
4	A	202201	2
5	A	202202	3
6	A	202203	4
7	B	202110	2
8	B	202111	4
9	B	202112	8
10	B	202201	4

M

Note that we don't need to list the columns that will remain fixed on each row in the pivoted table

A list of the items that will form the new column headers

```
Table.Pivot(
    df,
    List.Distinct(df[YearMonth]),
    "YearMonth",
    "Value",
    List.Sum
)
```

The unpivoted data as a table

The column in which we'll find the items for the column headers

The column from which to take the values for the new columns

The function to apply to the values for the new columns

Python

If we know there's exactly one value for each intersection of YearMonth and Item, we can use df.pivot and omit the "aggfunc" parameter.

The unpivoted data as a DataFrame

The column(s) to keep fixed on each row

```
df.pivot_table(
    index='Item',
    columns='YearMonth',
    values='Value',
    aggfunc='sum'
)
```

The column in which we'll find the items for the column headers

The column from which to take the values for the new columns

The function to apply to the values for the new columns

R

The unpivoted data as a DataFrame

The column in which we'll find the items for the column headers

```
dcast(df, Item ~ YearMonth, value.var = "Value")
```

The column(s) to keep fixed on each row

The column from which to take the values for the new columns

The dcast function is in the data.table package

Takeaways:

1. Unpivoting data is a common need in data analytics

2. Understanding how to do it in one language can make learning another much easier

3. In Python, we can use df.pivot_table if we need to apply an aggregate, or df.pivot if we don't

4. In R, we can unpivot using melt from either reshape2 or data.table