Wide and long data formats

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

You will learn

- Wide and long formats
- Long to wide transformation
- Wide to long transformation
- Stacking and unstacking columns
- Reshaping and handling complex data, such as string columns or JSON data

Why it is important

- Tidy datasets
- Data is not in the appropriate format for analysis:
  - Human readable vs. statistical analysis
- Nested data in DataFrames is complex to handle
- Get summary statistics for multi-level index DataFrames

Shape of data

- The way in which a dataset is organized in rows and columns

```
fifa_players = pd.read_csv("fifa_players.csv")
fifa_players
```

```
                name    age   nationality           club
0       Lionel Messi     32     Argentina      Barcelona
1  Cristiano Ronaldo     34      Portugal       Juventus
2    Neymar da Silva     27        Brazil  Saint-Germain
```

```
fifa_players.shape
```

```
(3, 4)
```

Wide format

```
fifa_players
```

```
                name    age   nationality           club
0       Lionel Messi     32     Argentina      Barcelona
1  Cristiano Ronaldo     34      Portugal       Juventus
2    Neymar da Silva     27        Brazil  Saint-Germain
```

Wide format

```
fifa_players
```

```
                          vv
                name    | age | nationality          club
0       Lionel Messi    | 32  |  Argentina      Barcelona
1  Cristiano Ronaldo    | 34  |   Portugal       Juventus
2    Neymar da Silva    | 27  |     Brazil  Saint-Germain
                          ^^
```

- Each feature is in a separate column

Wide format

```
fifa_players
```

```
                name    age   nationality           club
0       Lionel Messi     32     Argentina      Barcelona <--
1  Cristiano Ronaldo     34      Portugal       Juventus <--
2    Neymar da Silva     27        Brazil  Saint-Germain <--
```

- Each feature is in a separate column
- Each rows contains many features of the same player

Wide format

```
fifa_players
```

```
                name    age   nationality           club
0       Lionel Messi     32     Argentina      Barcelona
 ---------------------------------------------------------
1    Cristiano Ronaldo    NaN  <-    Portugal       Juventus
 ---------------------------------------------------------
2    Neymar da Silva     27        Brazil  Saint-Germain
```

- Each feature is in a separate column
- Each rows contains many features of the same player
- No repetition but large number of missing values
- Simple statistics and imputation

Long format

```
fifa_players_long.head()
```

```
                name     variable     value
0 Cristiano Ronaldo  nationality   Portugal
1 Cristiano Ronaldo         club   Juventus
2      Lionel Messi          age         32
3      Lionel Messi  nationality  Argentina
4      Lionel Messi         club  Barcelona
```

Long format

```
fifa_players_long.head()
```

```
                name     variable     value
0   Cristiano Ronaldo  nationality   Portugal <--
1 Cristiano Ronaldo         club  Juventus
2      Lionel Messi          age        32
3        Lionel Messi  nationality  Argentina <--
4      Lionel Messi         club  Barcelona
```

- Each row represents one feature

Long format

```
fifa_players_long.head()
```

```
                name     variable     value
0   Cristiano Ronaldo  nationality   Portugal <--
1   Cristiano Ronaldo         club   Juventus <--
2      Lionel Messi          age         32
3      Lionel Messi  nationality  Argentina
4      Lionel Messi         club  Barcelona
```

- Each row represents one feature
- Multiple rows for each player

Long format

```
fifa_players_long.head()
```

```
  |              name |    variable      value
0 | Cristiano Ronaldo | nationality   Portugal
1 | Cristiano Ronaldo |        club   Juventus
2 |      Lionel Messi |         age         32
3 |      Lionel Messi | nationality  Argentina
4 |      Lionel Messi |        club  Barcelona
        ^^^^^^^^^^^
```

- Each row represents one feature
- Multiple rows for each player
- A column (`name`) to identify same player

Long format

```
fifa_players_long.head()
```

```
                name     variable     value
0 Cristiano Ronaldo  nationality   Portugal
1 Cristiano Ronaldo         club   Juventus
2      Lionel Messi          age         32
3      Lionel Messi  nationality  Argentina
4      Lionel Messi         club  Barcelona
```

- Each row represents one feature
- Multiple rows for each player
- A column (`name`) to identify same player
- Tidy data:
  - Better to summarize data
  - Key-value pairs
  - Preferred for analysis and graphing

Reshaping data

- Transforming a DataFrame or Series structure to adjust it for analysis
  - Transposing a DataFrame

```
fifa_players.set_index('club')
```

```
                          name  age  nationality
         club
    Barcelona        Lionel Messi   32    Argentina
     Juventus   Cristiano Ronaldo  NaN     Portugal
Saint-Germain     Neymar da Silva   27       Brazil
```

Reshaping data

- Transforming a DataFrame or Series structure to adjust it for analysis
  - Transposing a DataFrame

```
fifa_players.set_index('club')[['name', 'nationality']]
```

```
                              name   nationality
         club
    Barcelona         Lionel Messi       Argentina
     Juventus    Cristiano Ronaldo        Portugal
Saint-Germain      Neymar da Silva          Brazil
```

Reshaping data

- Transforming a DataFrame or Series structure to adjust it for analysis
  - Transposing a DataFrame

```
fifa_players.set_index('club')[['name', 'nationality']].transpose()
```

```
       club         Barcelona           Juventus     Saint-Germain
       name      Lionel Messi  Cristiano Ronaldo   Neymar da Silva
nationality         Argentina           Portugal            Brazil
```

Reshaping data

- Converting data from wide to long format and vice versa
- Unit of analysis:
  - Long format -> characteristic of a player
  - Wide format -> each player

Wide to long transformation

- Performed using `pandas` functions, such as:
  - `.melt()`
  - `.wide_to_long()`

Long to wide format

- Transform data using `pandas` methods, for example:
  - `.pivot()`
  - `.pivot_table()`

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Reshaping using pivot method

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

From long to wide

- Demonstrate relationship between two columns
- Time series operations with the variables
- Operation that requires columns to be the unique variable

[1] https://pandas.pydata.org/docs/user_guide/reshaping.html

From long to wide

	Name	Year	Weight
0	John	2013	80
1	Mary	2013	65
2	Mary	2014	68
3	John	2014	83
4	Laura	2014	71

Pivot method

	Name	Year	Weight
0	John	2013	80
1	Mary	2013	65
2	Mary	2014	68
3	John	2014	83
4	Laura	2014	71

Name	John	Mary	Laura
Year			
2013	80	65	NaN
2014	83	68	71

```
df.pivot(                    ,                    )
```

Pivot method

	Name	Year	Weight
0	John	2013	80
1	Mary	2013	65
2	Mary	2014	68
3	John	2014	83
4	Laura	2014	71

Name	John	Mary	Laura
Year			
2013	80	65	NaN
2014	83	68	71

```
df.pivot(index=          , columns=          , values=          )
```

Pivot method

	Name	Year	Weight
0	John	2013	80
1	Mary	2013	65
2	Mary	2014	68
3	John	2014	83
4	Laura	2014	71

Name	John	Mary	Laura
Year			
2013	80	65	NaN
2014	83	68	71

```
df.pivot(index="Year", columns=          , values=          )
```

Pivot method

	Name	Year	Weight
0	John	2013	80
1	Mary	2013	65
2	Mary	2014	68
3	John	2014	83
4	Laura	2014	71

Name	John	Mary	Laura
Year			
2013	80	65	NaN
2014	83	68	71

```
df.pivot(index="Year", columns="Name", values=          )
```

Pivot method

	Name	Year	Weight
0	John	2013	80
1	Mary	2013	65
2	Mary	2014	68
3	John	2014	83
4	Laura	2014	71

Name	John	Mary	Laura
Year			
2013	80	65	NaN
2014	83	68	71

```
df.pivot(index="Year", columns="Name", values="Weight")
```

Pivot method

	Name	Year	Weight
0	John	2013	80
1	Mary	2013	65
2	Mary	2014	68
3	John	2014	83
4	Laura	2014	71

Name	John	Mary	Laura
Year			
2013	80	65	NaN
2014	83	68	71

```
df.pivot(index="Year", columns="Name", values="Weight")
```

Pivoting a dataset

```
fifa = pd.read_csv('fifa_players.csv')
fifa.head()
```

```
                 name    variable  metric_system  imperial_system
0   Cristiano Ronaldo      weight             83           183.00
1            J. Oblak      weight             87           191.00
2   Cristiano Ronaldo      height            187             6.13
3     J. Oblak             height            188             6.16
```

Pivoting a dataset

```
fifa.pivot(index='name'                                                    )
```

Pivoting a dataset

```
fifa.pivot(index='name', columns='variable'                          )
```

Pivoting a dataset

```
fifa.pivot(index='name', columns='variable', values='metric_system')
```

```
         variable  height   weight
             name
Cristiano Ronaldo     187       83
         J. Oblak     188       87
```

Pivoting multiple columns

```
fifa.pivot(index='name', columns='variable', values=['metric_system', 'imperial_system'])
```

```
                    metric_system     imperial_system
         variable   height  weight     height   weight
             name
Cristiano Ronaldo      187      83       6.13    183.0
         J. Oblak      188      87       6.16    191.0
```

Pivoting multiple columns

	Name	Year	Weight	Age
0	John	2013	80	30
1	Mary	2013	65	28
2	Mary	2014	68	29
3	John	2014	83	31
4	Laura	2014	71	34

	Weight			Age		
Name	John	Mary	Laura	John	Mary	Laura
Year						
2013	80	65	NaN	30	28	NaN
2014	83	68	71	31	29	34

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

Pivoting multiple columns

```
fifa.pivot(index="name", columns="variable")
```

```
                     metric_system     imperial_system
         variable   height  weight     height   weight
             name
Cristiano Ronaldo      187      83       6.13    183.0
         J. Oblak      188      87       6.16    191.0
```

Duplicate entries error

```
another_fifa.head()
```

```
                name    variable  metric_system  imperial_system
0   Cristiano Ronaldo      weight             83           183.00
1            J. Oblak      weight             87           191.00
2   Cristiano Ronaldo      height            187             6.13
3            J. Oblak      height            188             6.16
4   Cristiano Ronaldo      height            187             6.14
```

Duplicate entries error

```
another_fifa.head()
```

```
                name    variable  metric_system  imperial_system
0   Cristiano Ronaldo      weight             83           183.00
1            J. Oblak      weight             87           191.00
  2   Cristiano Ronaldo      height            187             6.13 <--
3            J. Oblak      height            188             6.16
  4   Cristiano Ronaldo      height            187             6.14 <--
```

Duplicate entries error

```
another_fifa.pivot(index="name", columns="variable")
```

```
ValueError: Index contains duplicate entries, cannot reshape
```

```
another_fifa = another_fifa.drop(4, axis=0)
another_fifa.pivot(index="name", columns="variable")
```

```
                    metric_system     imperial_system
         variable   height  weight     height   weight
             name
Cristiano Ronaldo      187      83       6.13    183.0
         J. Oblak      188      87       6.16    191.0
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Pivot tables

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Pivot method limitations

```
another_fifa.head()
```

```
                name    variable  metric_system  imperial_system
0   Cristiano Ronaldo      weight             83           183.00
1            J. Oblak      weight             87           191.00
2   Cristiano Ronaldo      height            187             6.13
3     J. Oblak             height            188             6.16
4   Cristiano Ronaldo      height            187             6.14
```

```
another_fifa.pivot(index="name", columns="variable")
```

```
Traceback (most recent call last):
   ValueError: Index contains duplicate entries, cannot reshape
```

Pivot method limitations

- General purpose pivoting
- Index/column pair must be unique
- Cannot aggregate values

Pivot table

- A DataFrame containing statistics that summarizes the data of a larger DataFrame

Name	John	Mary
Year		
2013	80.5	66.5
2014	83	68

Pivot table

	Name	Year	Weight
0	John	2013	80
1	John	2013	81
2	Mary	2013	67
3	Mary	2013	66
4	John	2014	82
5	John	2014	84
6	Mary	2014	69
7	Mary	2014	67

Pivot table

	Name	Year	Weight
0	John	2013	80
1	John	2013	81
2	Mary	2013	67
3	Mary	2013	66
4	John	2014	82
5	John	2014	84
6	Mary	2014	69
7	Mary	2014	67

Name	John	Mary
Year		
2013	80.5	66.5
2014	83	68

```
df.pivot_table(                    ,                    ,                    )
```

Pivot table

	Name	Year	Weight
0	John	2013	80
1	John	2013	81
2	Mary	2013	67
3	Mary	2013	66
4	John	2014	82
5	John	2014	84
6	Mary	2014	69
7	Mary	2014	67

Name	John	Mary
Year		
2013	80.5	66.5
2014	83	68

```
df.pivot_table(index="Year", columns="Name",                    ,                    )
```

Pivot table

	Name	Year	Weight
0	John	2013	80
1	John	2013	81
2	Mary	2013	67
3	Mary	2013	66
4	John	2014	82
5	John	2014	84
6	Mary	2014	69
7	Mary	2014	67

Name	John	Mary
Year		
2013	80.5	66.5
2014	83	68

```
df.pivot_table(index="Year", columns="Name", values="Weight", aggfunc="mean")
```

Pivot table

```
another_fifa.pivot_table(index="name", columns="variable", aggfunc="mean")
```

```
                    metric_system     imperial_system
         variable   height  weight     height   weight
             name
Cristiano Ronaldo      187      83      6.135    183.0
         J. Oblak      188      87      6.160    191.0
```

Hierarchical indexes

```
fifa_players.head(6)
```

```
      first     last   movement   overall  attacking
0    Lionel    Messi   shooting        92         70
1 Cristiano  Ronaldo   shooting        93         89
2    Lionel    Messi    passing        92         92
3 Cristiano  Ronaldo    passing        82         83
4    Lionel    Messi    passing        96         88
5 Cristiano  Ronaldo    passing        89         84
```

Hierarchical indexes

```
fifa_players.head(6)
```

```
      first     last   movement   overall  attacking
0    Lionel    Messi   shooting        92         70
1 Cristiano  Ronaldo   shooting        93         89
2    Lionel    Messi    passing        92         92
3 Cristiano  Ronaldo    passing        82         83
4    Lionel    Messi    passing        96         88
5 Cristiano  Ronaldo    passing        89         84
```

```
fifa_players.pivot_table(index=                    , columns="movement", values=                    , aggfunc=     )
```

Hierarchical indexes

```
fifa_players.head(6)
```

```
      first     last   movement   overall  attacking
0    Lionel    Messi   shooting        92         70
1 Cristiano  Ronaldo   shooting        93         89
2    Lionel    Messi    passing        92         92
3 Cristiano  Ronaldo    passing        82         83
4    Lionel    Messi    passing        96         88
5 Cristiano  Ronaldo    passing        89         84
```

```
fifa_players.pivot_table(index=["first", "last"], columns="movement", values=                    , aggfunc=     )
```

Hierarchical indexes

```
fifa_players.head(6)
```

```
      first     last   movement   overall  attacking
0    Lionel    Messi   shooting        92         70
1 Cristiano  Ronaldo   shooting        93         89
2    Lionel    Messi    passing        92         92
3 Cristiano  Ronaldo    passing        82         83
4    Lionel    Messi    passing        96         88
5 Cristiano  Ronaldo    passing        89         84
```

```
fifa_players.pivot_table(index=["first", "last"], columns="movement", values=["overall", "attacking"], aggfunc="max")
```

```
                          attacking           overall
          movement  passing shooting  passing shooting
    first     last
Cristiano  Ronaldo       84       89       89       93
   Lionel    Messi       92       70       96       92
```

Margins

```
fifa_players.pivot_table(index=["first", "last"], columns="movement", aggfunc="count",            )
```

Margins

```
fifa_players.pivot_table(index=["first", "last"], columns="movement", aggfunc="count", margins=True)
```

```
                              attacking                  overall
          movement  passing shooting  All    passing shooting  All
    First     Last
Cristiano  Ronaldo        2        1    3          2        1    3
   Lionel    Messi        2        1    3          2        1    3
      All                 4        2    6          4        2    6
```

Pivot or pivot table?

Does the DataFrame have more than one value for each index/column pair?

Do you need to have a multi-index in your resulting pivoted DataFrame?

Do you need summary statistics of your large DataFrame?

Yes! Use `.pivot_table()`

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Reshaping with melt

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Wide to long transformation

- Perform analytics
- Plot different variables in the same graph

Wide to long transformation

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	variable	value
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155
6	John	Wick	weight	70
7	Mary	Shelley	weight	60
8	Alice	Liddell	weight	58

```
df.melt(                                    )
```

Melt

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	variable	value
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155
6	John	Wick	weight	70
7	Mary	Shelley	weight	60
8	Alice	Liddell	weight	58

```
df.melt(id_vars=                    )
```

Melt

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	variable	value
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155
6	John	Wick	weight	70
7	Mary	Shelley	weight	60
8	Alice	Liddell	weight	58

```
df.melt(id_vars=["first", "last"])
```

Melt

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	variable	value
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155
6	John	Wick	weight	70
7	Mary	Shelley	weight	60
8	Alice	Liddell	weight	58

```
df.melt(id_vars=["first", "last"])
```

Melt

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	variable	value
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155
6	John	Wick	weight	70
7	Mary	Shelley	weight	60
8	Alice	Liddell	weight	58

```
df.melt(id_vars=["first", "last"])
```

Melting data

```
books
```

```
                            title isbn language pages
0                  Mostly Harmless  074      eng   260
1           The Hitchhiker's Guide  072      eng   215
2 El restaurante del fin del mundo  071      spa   250
```

```
books.melt(id_vars='title')
```

```
                              title  variable   value
0                    Mostly Harmless      isbn     074
1             The Hitchhiker's Guide      isbn     072
2  El restaurante del fin del mundo      isbn     071
3                    Mostly Harmless  language     eng
4             The Hitchhiker's Guide  language     eng
5  El restaurante del fin del mundo  language     spa
6                    Mostly Harmless     pages     260
7             The Hitchhiker's Guide     pages     215
8  El restaurante del fin del mundo     pages     250
```

Values and variables

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	feature	amount
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155

```
df.melt(id_vars=["first","last"], value_vars=          , var_name=          , value_name=          )
```

Values and variables

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	feature	amount
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155

```
df.melt(id_vars=["first","last"], value_vars=["age","height"], var_name=          , value_name=          )
```

Values and variables

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	feature	amount
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155

```
df.melt(id_vars=["first","last"], value_vars=["age","height"], var_name="feature", value_name="amount")
```

Values and variables

	first	last	age	height	weight
0	John	Wick	50	185	70
1	Mary	Shelley	25	164	60
2	Alice	Liddell	16	155	58

	first	last	feature	amount
0	John	Wick	age	50
1	Mary	Shelley	age	25
2	Alice	Liddell	age	16
3	John	Wick	height	185
4	Mary	Shelley	height	164
5	Alice	Liddell	height	155

```
df.melt(id_vars=["first", "last"], value_vars=["age", "height"], var_name="feature", value_name="amount")
```

Specifying values to melt

```
books.melt(id_vars='title', value_vars=['language_code', 'num_pages'])
```

```
                              title  variable   value
0                   Mostly Harmless  language     eng
1             The Hitchhiker's Guide  language     eng
2  El restaurante del fin del mundo  language     spa
3                   Mostly Harmless     pages     260
4             The Hitchhiker's Guide     pages     215
5  El restaurante del fin del mundo     pages     250
```

Naming values and variables

```
books.melt(id_vars='title', value_vars=['language_code', 'isbn'], var_name='feature', value_name='code')
```

```
                              title   feature    code
0                   Mostly Harmless      isbn     074
1            The Hitchhiker's Guide      isbn     072
2  El restaurante del fin del mundo      isbn     071
3                   Mostly Harmless  language     eng
4            The Hitchhiker's Guide  language     eng
5  El restaurante del fin del mundo  language     spa
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Wide to long function

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Wide to long transformation

	name	age2019	weight2019	age2020	weight2020
0	John Wick	50	70	51	72
1	Mary Shelley	25	60	26	61
2	Alice Liddell	16	58	17	57

Wide to long transformation

	name	age2019	weight2019	age2020	weight2020
0	John Wick	50	70	51	72
1	Mary Shelley	25	60	26	61
2	Alice Liddell	16	58	17	57

		age	weight
name	year		
John Wick	2019	50	70
Mary Shelley	2019	25	60
Alice Liddell	2019	16	58
John Wick	2020	51	72
Mary Shelley	2020	26	61
Alice Liddell	2020	17	57

```
pd.wide_to_long(                                                    )
```

Wide to long function

	name	age2019	weight2019	age2020	weight2020
0	John Wick	50	70	51	72
1	Mary Shelley	25	60	26	61
2	Alice Liddell	16	58	17	57

		age	weight
name	year		
John Wick	2019	50	70
Mary Shelley	2019	25	60
Alice Liddell	2019	16	58
John Wick	2020	51	72
Mary Shelley	2020	26	61
Alice Liddell	2020	17	57

```
pd.wide_to_long(df, stubnames =          , i =          , j =          )
```

Wide to long function

	name	age2019	weight2019	age2020	weight2020
0	John Wick	50	70	51	72
1	Mary Shelley	25	60	26	61
2	Alice Liddell	16	58	17	57

		age	weight
name	year		
John Wick	2019	50	70
Mary Shelley	2019	25	60
Alice Liddell	2019	16	58
John Wick	2020	51	72
Mary Shelley	2020	26	61
Alice Liddell	2020	17	57

```
pd.wide_to_long(df , stubnames = ["age", "weight"] , i =          , j =          )
```

Wide to long function

	name	age2019	weight2019	age2020	weight2020
0	John Wick	50	70	51	72
1	Mary Shelley	25	60	26	61
2	Alice Liddell	16	58	17	57

		age	weight
name	year		
John Wick	2019	50	70
Mary Shelley	2019	25	60
Alice Liddell	2019	16	58
John Wick	2020	51	72
Mary Shelley	2020	26	61

```
pd.wide_to_long(df , stubnames = ["age", "weight"] , i =          , j = "year" )
```

Wide to long function

	name	age2019	weight2019	age2020	weight2020
0	John Wick	50	70	51	72
1	Mary Shelley	25	60	26	61
2	Alice Liddell	16	58	17	57

		age	weight
name	year		
John Wick	2019	50	70
Mary Shelley	2019	25	60
Alice Liddell	2019	16	58
John Wick	2020	51	72
Mary Shelley	2020	26	61
Alice Liddell	2020	17	57

```
pd.wide_to_long(df , stubnames = ["age", "weight"] , i = "name" , j = "year" )
```

Reshaping data

```
books
```

```
                              title ratings2019 sold2019 ratings2020 sold2020
0                   Mostly Harmless         4.2      456         4.3      436
1             The Hitchhiker's Guide         4.8      980         4.9      998
2 El restaurante del fin del mundo         4.5      678         4.6      638
```

Reshaping data

```
pd.wide_to_long(books,                                                    )
```

Reshaping data

```
pd.wide_to_long(books, stubnames=['ratings', 'sold']                    )
```

Reshaping data

```
pd.wide_to_long(books, stubnames=['ratings', 'sold'],          , j='year')
```

Reshaping data

```
pd.wide_to_long(books, stubnames=['ratings', 'sold'], i='title', j='year')
```

```
                                      ratings    sold
                            title  year
0                  Mostly Harmless  2019     4.2     456
1            The Hitchhiker's Guide  2019     4.8     980
2 El restaurante del fin del mundo  2019     4.5     678
3                  Mostly Harmless  2020     4.4     436
4            The Hitchhiker's Guide  2020     4.9     998
5 El restaurante del fin del mundo  2020     4.6     638
```

DataFrame with index

```
books_with_index
```

```
                                  author  ratings2019 sold2019
                    title
0   To Kill a Mockingbird       Harper Lee          4.7      456
1  The Hitchhiker's Guide    Douglas Adams          4.8      980
2           The Black Cat   Edgar Alan Poe          4.5      678
```

```
pd.wide_to_long(books_with_index, stubnames=['ratings', 'sold'], i='author', j='year')
```

```
                        ratings    sold
          author  year
0     Harper Lee  2019      4.2      456
1  Douglas Adams  2019      4.8      980
2 Edgar Alan Poe  2019      4.5      678
```

DataFrame with index

```
books_with_index.reset_index(drop=False, inplace=True)
pd.wide_to_long(books_with_index, stubnames=['ratings', 'sold'], i=['author', 'title'], j='year')
```

```
                                                ratings  sold
                    title           author    year
0   To Kill a Mockingbird       Harper Lee    2019       4.7   456
1  The Hitchhiker's Guide    Douglas Adams    2019       4.8   980
2           The Black Cat   Edgar Alan Poe    2019       4.5   678
```

sep argument

```
new_books
```

```
                  title              author  ratings_2019 sold_2019 ratings_2020 sold_2020
0 A Murder Is Announced     Agatha Christie           4.4       796           4.8       856
1       Sherlock Holmes  Sir A. Conan Doyle           4.5       780           4.8       818
2           The Sparrow  Mary Doria Russell           4.2       178           4.1       238
```

sep argument

```
pd.wide_to_long(new_books, stubnames=['ratings', 'sold'], i=['title', 'author'], j='year')
```

```
                    sold_2020 ratings_2020 ratings_2019 sold_2019  ratings sold
title  author year
```

sep argument

```
pd.wide_to_long(new_books, stubnames=['ratings', 'sold'], i=['title', 'author'], j='year', sep='_')
```

```
                                                ratings    sold
                   title              author  year
0  A Murder Is Announced     Agatha Christie  2019     4.4     796
1        Sherlock Holmes  Sir A. Conan Doyle  2019     4.5     780
2            The Sparrow  Mary Doria Russell  2019     4.2     178
3  A Murder Is Announced     Agatha Christie  2020     4.8     856
4        Sherlock Holmes  Sir A. Conan Doyle  2020     4.8     818
5            The Sparrow  Mary Doria Russell  2020     4.1     238
```

suffix argument

```
another_books
```

```
                  title  ratings_one  sold_one  ratings_two  sold_two
0 A Murder Is Announced          4.4       796          4.8       856
1       Sherlock Holmes          4.5       780          4.8       818
2           The Sparrow          4.2       178          4.1       238
```

suffix argument

```
pd.wide_to_long(another_books, stubnames=['ratings', 'sold'], i='title', j='edition', sep='_')
```

```
            sold_one ratings_one ratings_two sold_two  ratings sold
title  year
```

suffix argument

```
pd.wide_to_long(another_books, stubnames=['ratings', 'sold'], i='title', j='edition', sep='_', suffix='\w+')
```

```
                                ratings    sold
                   title edition
0  A Murder Is Announced     one     4.4     796
1        Sherlock Holmes     one     4.5     780
2            The Sparrow     one     4.2     178
3  A Murder Is Announced     two     4.8     856
4        Sherlock Holmes     two     4.8     818
5            The Sparrow     two     4.1     238
```

Let's practice!

RESHAPING DATA WITH PANDAS

Working with string columns

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Columns with strings

```
books
```

```
                    title  raitings_2015 sold_2015 raitings_2016 sold_2016
0  The Civil War:Vol. 1             4.3       234           4.2       254
1  The Civil War:Vol. 2             4.5       525           4.3       515
2  The Civil War:Vol. 3             4.1       242           4.2       251
```

```
books['title'].dtypes
```

```
dtype('O')
```

String methods

- `pandas` Series string processing methods
- Access easily by `str` attribute

Splitting into two columns

```
books
```

```
                    title  raitings_2015 sold_2015 raitings_2016 sold_2016
0  The Civil War:Vol. 1             4.3       234           4.2       254
1  The Civil War:Vol. 2             4.5       525           4.3       515
2  The Civil War:Vol. 3             4.1       242           4.2       251
```

```
books['title']
```

Splitting into two columns

```
books
```

```
                    title  raitings_2015 sold_2015 raitings_2016 sold_2016
0  The Civil War:Vol. 1             4.3       234           4.2       254
1  The Civil War:Vol. 2             4.5       525           4.3       515
2  The Civil War:Vol. 3             4.1       242           4.2       251
```

```
books['title'].str.split(':')
```

```
0    [The Civil War,  Vol. 1]
1    [The Civil War,  Vol. 2]
2    [The Civil War,  Vol. 3]
```

Splitting into two columns

```
books
```

```
                    title  raitings_2015 sold_2015 raitings_2016 sold_2016
0  The Civil War:Vol. 1             4.3       234           4.2       254
1  The Civil War:Vol. 2             4.5       525           4.3       515
2  The Civil War:Vol. 3             4.1       242           4.2       251
```

```
books['title'].str.split(":").str.get(0)
```

```
0    The Civil War
1    The Civil War
2    The Civil War
```

Splitting into two columns

```
books
```

```
                    title  raitings_2015 sold_2015 raitings_2016 sold_2016
0  The Civil War:Vol. 1             4.3       234           4.2       254
1  The Civil War:Vol. 2             4.5       525           4.3       515
2  The Civil War:Vol. 3             4.1       242           4.2       251
```

```
books['title'].str.split(":", expand=True)
```

```
                0         1
0  The Civil War   Vol. 1
1  The Civil War   Vol. 2
2  The Civil War   Vol. 3
```

Splitting into two columns

```
books[['main_title', 'subtitle']] = books['title'].str.split(":", expand=True)
```

```
books.drop('title', axis=1, inplace=True)
pd.wide_to_long(books , stubnames=['ratings', 'sold'], i=['main_title', 'subtitle'], j='year')
```

```
                                ratings sold
   main_title  subtitle  year
The Civil War    Vol. 1  2015      4.3  234
                         2016      4.2  254
                 Vol. 2  2015      4.5  525
                         2016      4.3  515
                 Vol. 3  2015      4.1  242
                         2016      4.2  251
```

Concatenate two columns

```
books_new
```

```
   name_author  lastname_author  nationality  number_books
0    Virginia             Wolf      British            50
1    Margaret           Atwood     Canadian            40
2      Harper              Lee     American             2
```

Concatenate two columns

```
books_new
```

```
   name_author  lastname_author  nationality  number_books
0    Virginia             Wolf      British            50
1    Margaret           Atwood     Canadian            40
2      Harper              Lee     American             2
```

```
books_new['name_author'].str.cat(books_new['lastname_author'], sep=' ')
```

```
0      Virginia Wolf
1    Margaret Atwood
2         Harper Lee
```

Concatenate two columns

```
books_new
```

```
   name_author  lastname_author  nationality  number_books
0    Virginia             Wolf      British            50
1    Margaret           Atwood     Canadian            40
2      Harper              Lee     American             2
```

```
books_new['author'] = books_new['name_author'].str.cat(books_new['lastname_author'], sep=' ')
books_new
```

```
  name_author lastname_author nationality  number_books           author
0    Virginia            Wolf     British            50    Virginia Wolf
1    Margaret          Atwood    Canadian            40  Margaret Atwood
2      Harper             Lee    American             2       Harper Lee
```

Concatenate two columns

```
books_new
```

```
   name_author  lastname_author  nationality  number_books
0    Virginia             Wolf      British            50
1    Margaret           Atwood     Canadian            40
2      Harper              Lee     American             2
```

```
books_new.melt(id_vars='author', value_vars=['nationality', 'number_books'], var_name='feature', value_name='value')
```

```
            author       feature     value
0    Virginia Wolf   nationality   British
1  Margaret Atwood   nationality  Canadian
2       Harper Lee   nationality  American
3    Virginia Wolf  number_books        50
4  Margaret Atwood  number_books        40
5       Harper Lee  number_books         2
```

Concatenate index

```
comics_marvel
```

```
           subtitle  year  ratings  sold
main_title
Avengers       Next  1992      4.5   234
Avengers    Forever  1998      4.6   224
Avengers       2099  1999      4.8   141
```

Concatenate index

```
comics_marvel.head(2)
```

```
           subtitle  year  ratings  sold
main_title
Avengers       Next  1992      4.5   234
Avengers    Forever  1998      4.6   224
```

```
comics_marvel.index = comics_marvel.index.str.cat(comics_marvel['subtitle'], sep='-')
books
```

```
                 subtitle  year  ratings  sold
main_title
Avengers-Next        Next  1992      4.5   234
Avengers-Forever  Forever  1998      4.6   224
Avengers-2099        2099  1999      4.8   141
```

Split index

```
comics_marvel.index = comics_marvel.index.str.split('-', expand=True)
comics_marvel
```

```
                  subtitle  year  ratings  sold
Avengers Next        Next  1992      4.5   234
         Forever  Forever  1998      4.6   224
         2099        2099  1999      4.8   141
```

Concatenate Series

```
books_new['name_author']
```

```
0    Virginia
1    Margaret
2      Harper
```

```
new_list = ['Wolf', 'Atwood', 'Lee']
books_new['name_author'].str.cat(new_list, sep=' ')
```

```
0      Virginia Wolf
1    Margaret Atwood
2         Harper Lee
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Stacking DataFrames

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Row multi-indices

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

Setting the index

```
churn
```

```
   credit_score age  country  num_products exited
0           619  43   France             1    Yes
1           608  34  Germany             0     No
2           502  23   France             1    Yes
```

Setting the index

```
churn.set_index(['country', 'age'], inplace=True)
```

```
              credit_score num_products exited
age  country
 43   France           619            1    Yes
 34  Germany           608            0     No
 23   France           502            1    Yes
```

MultiIndex from array

```
new_array = [['yes', 'no', 'yes'], ['no', 'yes', 'yes']]
churn.index = pd.MultiIndex.from_arrays(new_array, names=['member', 'credit_card'])
churn
```

```
                    credit_score age  country  num_products exited
member credit_card
   yes          no           619  43   France             1    Yes
    no         yes           608  34  Germany             0     No
   yes         yes           502  23   France             1    Yes
```

MultiIndex DataFrames

		2019		2020	
		height	weight	height	weight
Last	First				
Wick	John	185	68	185	70
	Julien	164	61	164	60
Shelley	Mary	164	59	164	60
	Frank	155	65	155	58

MultiIndex DataFrames

```
index = pd.MultiIndex.from_arrays([['Wick', 'Wick', 'Shelley', 'Shelley'],
                                   ['John', 'Julien', 'Mary', 'Frank']],
                        names=['last', 'first'])
columns = pd.MultiIndex.from_arrays([['2019', '2019', '2020', '2020'],
                                     ['age', 'weight', 'age', 'weight']],
                          names=['year', 'feature'])
patients = pd.DataFrame(data, index=index, columns=columns)
patients
```

```
            year        2019        2020
         feature  age weight  age weight
   last    first
   Wick     John   25     68   26     72
          Julien   31     72   32     73
Shelley     Mary   41     68   42     69
           Frank   32     75   33     74
```

The .stack() method

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

df.stack()

The .stack() method

> Rearrange a level of the columns to obtain a reshaped DataFrame with a new inner-most level row index

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

Stack into a series

```
churn
```

```
   credit_score age  country  num_products exited
0           619  43   France             1    Yes
1           608  34  Germany             0     No
2           502  23   France             1    Yes
```

```
churned_stacked = churn.stack()
churned_stacked.head(10)
```

```
member  credit_card
yes     no           credit_score        619
                     age                  43
                     country          France
                     num_products          1
                     churn               Yes
no      yes          credit_score        608
                     age                  34
                     country         Germany
                     num_products          0
                     churn                No
```

Stack into a DataFrame

```
patients
```

```
patients_stacked = patients.stack()
patients_stacked
```

```
            year        2019        2020
         feature  age weight  age weight
   last    first
   Wick     John   25     68   26     72
          Julien   31     72   32     73
Shelley     Mary   41     68   42     69
           Frank   32     75   33     74
```

```
                    year  2019 2020
  last   first  feature
  Wick    John      age    25   26
                 weight    68   72
        Julien      age    31   32
                 weight    72   73
Shelley  Mary       age    41   42
                 weight    68   69
        Frank       age    32   33
                 weight    75   74
```

Stack a level by number

```
patients
```

```
            year        2019        2020
         feature  age weight  age weight
   last    first
   Wick     John   25     68   26     72
          Julien   31     72   32     73
Shelley     Mary   41     68   42     69
           Frank   32     75   33     74
```

```
patients.stack(level=0)
```

```
                feature   age weight
   last  first    year
   Wick   John    2019    25     68
                  2020    26     72
        Julien    2019    31     72
                  2020    32     73
Shelley   Mary    2019    41     68
                  2020    42     69
         Frank    2019    32     75
                  2020    33     74
```

Stack a level by name

```
patients
```

```
            year        2019        2020
         feature  age weight  age weight
   last    first
   Wick     John   25     68   26     72
          Julien   31     72   32     73
Shelley     Mary   41     68   42     69
           Frank   32     75   33     74
```

```
patients.stack(level='year')
```

```
                feature   age weight
   last  first    year
   Wick   John    2019    25     68
                  2020    26     72
        Julien    2019    31     72
                  2020    32     73
Shelley   Mary    2019    41     68
                  2020    42     69
         Frank    2019    32     75
                  2020    33     74
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Unstacking DataFrames

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Review

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

df.stack()

Undoing stacking process

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

The .unstack() method

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

df.unstack()

The .unstack() method

> Rearrange a level of the row index into the columns to obtain a reshaped DataFrame with a new inner-most level column index.

Last	First		
Wick	John	height	185
		weight	68
	Julien	height	164
		weight	61
Shelley	Mary	height	164
		weight	59
	Frank	height	155
		weight	58

		height	weight
Last	First		
Wick	John	185	68
	Julien	164	61
Shelley	Mary	164	59
	Frank	155	58

Unstack Series

```
churn_stacked
```

```
member  credit_card
yes     no           credit_score        619
                     age                  43
                     country          France
                     num_products          1
                     churn               Yes
no      yes          credit_score        608
                     age                  34
                     country         Germany
                     num_products          0
                     churn                No
yes     yes          credit_score        502
                     age                  23
                     country          France
                     num_products          1
                     churn               Yes
```

Unstack Series

```
churned_stacked.unstack()
```

```
                    credit_score age  country  num_products exited
member credit_card
    no         yes           608  34  Germany             0     No
   yes          no           619  43   France             1    Yes
               yes           502  23   France             1    Yes
```

Unstacking a DataFrame

```
patients_stacked
```

```
                  year  2019 2020
  first   last feature
   Wick   John     age    25   26
                weight    68   72
        Julien     age    31   32
                weight    72   73
Shelley   Mary     age    41   42
                weight    68   69
         Frank     age    32   33
                weight    75   74
```

Unstacking a DataFrame

```
patients_stacked.unstack()
```

```
                      2019       2020
        feature  age weight age weight
   last   first
Shelley   Frank   32     75  33     74
           Mary   41     68  42     69
   Wick    John   25     68  26     72
         Julien   31     72  32     73
```

Unstack a level

Last	First		
Johnson	Louis	age	32
		weight	68
	Mary	age	42
		weight	61
Smith	Louis	age	20
		weight	59
	Mary	age	32
		weight	58

	First	Louis	Mary
Last			
Johnson	age	32	42
	weight	68	61
Smith	age	20	32
	weight	59	58

df.unstack(level=1) or df.unstack(level='First')

Unstack level by number

```
churn_stacked.head(10)
```

```
member  credit_card
yes     no           credit_score        619
                     age                  43
                     country          France
                     num_products          1
                     churn               Yes
no      yes          credit_score        608
                     age                  34
                     country         Germany
                     num_products          0
                     churn                No
```

```
churn_stacked.unstack(level=0)
```

```
                    member     no    yes
credit_card
         no credit_score     NaN    619
                     age     NaN     43
                 country     NaN France
            num_products     NaN      1
                   churn     NaN    Yes
        yes credit_score     608    502
                     age      34     23
                 country Germany France
            num_products       0      1
                   churn      No    Yes
```

Unstack level by name

```
churn_stacked.head(10)
```

```
member  credit_card
yes     no           credit_score        619
                     age                  43
                     country          France
                     num_products          1
                     churn               Yes
no      yes          credit_score        608
                     age                  34
                     country         Germany
                     num_products          0
                     churn                No
```

```
churn_stacked.unstack(level='credit_card')
```

```
                credit_card      no     yes
     member
         no credit_score     NaN     608
                     age     NaN      34
                 country     NaN Germany
            num_products     NaN       0
                   churn     NaN      No
        yes credit_score     619     NaN
                     age      43     NaN
                 country  France     NaN
            num_products       1     NaN
                   churn     Yes     NaN
```

Sort index

```
patients_stacked.unstack().sort_index(ascending=False)
```

```
           year         2019       2020
        feature   age weight age weight
   last   first
   Wick  Julien    31     72  32     73
           John    25     68  26     72
Shelley    Mary    41     68  42     69
          Frank    32     75  33     74
```

Rearranging levels

```
patients_stacked
```

```
                  year  2019 2020
  first   last feature
   Wick   John     age    25   26
                weight    68   72
        Julien     age    31   32
                weight    72   73
Shelley   Mary     age    41   42
                weight    68   69
         Frank     age    32   33
                weight    75   74
```

```
patients_stacked.unstack(level=1).stack(level=0)
```

```
first                  Frank  John  Julien  Mary
 last   feature year
Shelley age     2019   32.0   NaN     NaN  41.0
                2020   33.0   NaN     NaN  42.0
        weight  2019   75.0   NaN     NaN  68.0
                2020   74.0   NaN     NaN  69.0
Wick    age     2019    NaN  25.0    31.0   NaN
                2020    NaN  26.0    32.0   NaN
        weight  2019    NaN  68.0    72.0   NaN
                2020    NaN  72.0    73.0   NaN
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Working with multiple levels

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Instructor

datacamp

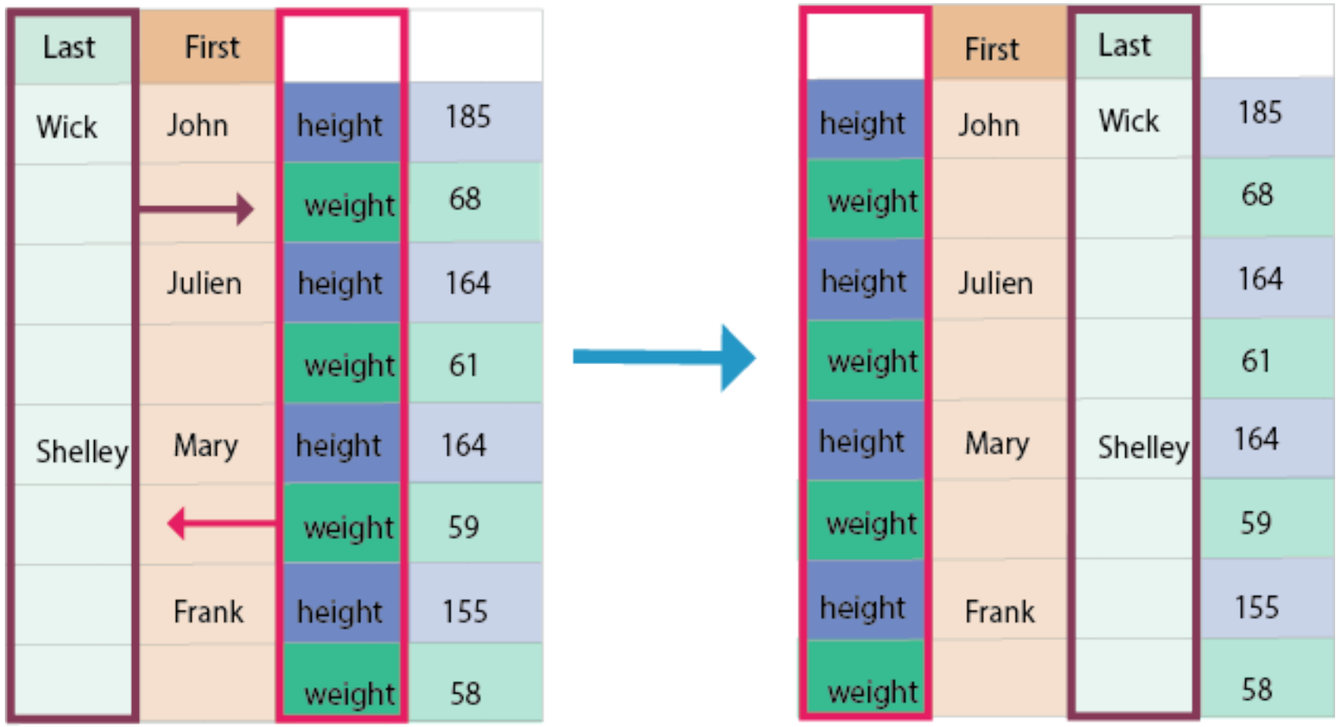
Review

- Stack and unstack DataFrames and Series
- Choose a level to stack or unstack by name or number
- Rearrange levels by combining unstack and stack

Rearranging multiple levels

- Swap levels
- Stack and unstack multiple levels at the same time

Swap levels



df.swaplevel(0 , 2)

Swap levels

```
cars
```

```
                          2019  2020
price    Golf       VW     25    26
 sold    Golf       VW     68    72
price  Passat       VW     31    32
 sold  Passat       VW     72    73
price A-class Mercedes     41    42
 sold A-class Mercedes     68    69
price C-class Mercedes     32    33
sold  C-class Mercedes     75    74
```

Swap levels

```
cars
```

```
                          2019  2020
price    Golf       VW     25    26
 sold    Golf       VW     68    72
price  Passat       VW     31    32
 sold  Passat       VW     72    73
price A-class Mercedes     41    42
 sold A-class Mercedes     68    69
price C-class Mercedes     32    33
sold  C-class Mercedes     75    74
```

```
cars.swaplevel(0, 2)
```

```
                          2019  2020
VW       Golf    price     25    26
                 sold      68    72
         Passat  price     31    32
                 sold      72    73
Mercedes A-class price     41    42
                 sold      68    69
         C-class price     32    33
                 sold      75    74
```

Swap levels and unstack

```
cars
```

```
                          2019  2020
price    Golf       VW     25    26
 sold    Golf       VW     68    72
price  Passat       VW     31    32
 sold  Passat       VW     72    73
price A-class Mercedes     41    42
 sold A-class Mercedes     68    69
price C-class Mercedes     32    33
sold  C-class Mercedes     75    74
```

```
cars.swaplevel(0, 2).unstack()
```

```
                          2019       2020
                     price sold price sold
Mercedes A-class        41   68    42   69
         C-class        32   75    33   74
     VW      Golf       25   68    26   72
              Passat    31   72    32   73
```

Swap levels and unstack

```
cars
```

```
                          2019  2020
price    Golf       VW     25    26
 sold    Golf       VW     68    72
price  Passat       VW     31    32
 sold  Passat       VW     72    73
price A-class Mercedes     41    42
 sold A-class Mercedes     68    69
price C-class Mercedes     32    33
sold  C-class Mercedes     75    74
```

```
cars.unstack()
```

```
              Mercedes    VW Mercedes    VW
                  2019  2019     2020  2020
price A-class     41.0   NaN     42.0   NaN
      C-class     32.0   NaN     33.0   NaN
         Golf      NaN  25.0      NaN  26.0
       Passat      NaN  31.0      NaN  32.0
sold  A-class     68.0   NaN     69.0   NaN
      C-class     75.0   NaN     74.0   NaN
         Golf      NaN  68.0      NaN  72.0
       Passat      NaN  72.0      NaN  73.0
```

Swap levels and unstack

```
cars
```

```
                          2019  2020
price    Golf       VW     25    26
 sold    Golf       VW     68    72
price  Passat       VW     31    32
 sold  Passat       VW     72    73
price A-class Mercedes     41    42
 sold A-class Mercedes     68    69
price C-class Mercedes     32    33
sold  C-class Mercedes     75    74
```

```
cars.unstack().swaplevel(0, 1, axis=1)
```

```
                          2019           2020
              Mercedes     VW Mercedes     VW
price A-class     41.0    NaN     42.0    NaN
      C-class     32.0    NaN     33.0    NaN
         Golf      NaN   25.0      NaN   26.0
       Passat      NaN   31.0      NaN   32.0
sold  A-class     68.0    NaN     69.0    NaN
      C-class     75.0    NaN     74.0    NaN
         Golf      NaN   68.0      NaN   72.0
       Passat      NaN   72.0      NaN   73.0
```

Swap levels and stack

```
cars
```

```
                          2019  2020
price    Golf       VW     25    26
 sold    Golf       VW     68    72
price  Passat       VW     31    32
 sold  Passat       VW     72    73
price A-class Mercedes     41    42
 sold A-class Mercedes     68    69
price C-class Mercedes     32    33
sold  C-class Mercedes     75    74
```

```
cars.stack()
```

```
price     Golf       VW   2019    25
                          2020    26
 sold     Golf       VW   2019    68
                          2020    72
price   Passat       VW   2019    31
                          2020    32
 sold   Passat       VW   2019    72
                          2020    73
price  A-class  Mercedes  2019    41
                          2020    42
 sold  A-class  Mercedes  2019    68
                          2020    69
```

Swap levels and stack

```
cars
```

```
                          2019  2020
price    Golf       VW     25    26
 sold    Golf       VW     68    72
price  Passat       VW     31    32
 sold  Passat       VW     72    73
price A-class Mercedes     41    42
 sold A-class Mercedes     68    69
price C-class Mercedes     32    33
sold  C-class Mercedes     75    74
```

```
cars.stack().swaplevel(0, 2)
```

```
     VW     Golf   price  2019    25
                           2020    26
                    sold  2019    68
                           2020    72
          Passat   price  2019    31
                           2020    32
                    sold  2019    72
                           2020    73
Mercedes  A-class  price  2019    41
                           2020    42
                    sold  2019    68
                           2020    69
```

Multiple levels

		day				night	
		2019		2020		2020	
		high	low	high	low	high	low
Last	First						
Wick	John	110	68	120	70	110	70
	Julien	120	61	121	60	115	60
Shelley	Mary	90	59	90	60	100	60
	Frank	100	65	92	58	105	58

Unstacking multiple levels

```
cars
```

```
year                        2019  2020
  brand      model  feature
     VW       Golf    price   25    26
                       sold   68    72
           Passat    price   31    32
                       sold   72    73
Mercedes  A-class    price   41    42
                       sold   68    69
          C-class    price   32    33
                       sold   75    74
```

Unstacking levels by number

```
cars.unstack(level=[0, 1])
```

```
                year            2019                          2020
brand             VW        Mercedes          VW         Mercedes
model   Golf Passat  A-class C-class Golf Passat  A-class C-class
feature
price    25     31        41      32   26     32       42      33
sold     68     72        68      75   72     73       69      74
```

Unstacking levels by name

```
cars.unstack(level=['brand', 'model'])
```

```
                year            2019                          2020
brand             VW        Mercedes          VW         Mercedes
model   Golf Passat  A-class C-class Golf Passat  A-class C-class
feature
price    25     31        41      32   26     32       42      33
sold     68     72        68      75   72     73       69      74
```

Stacking multiple levels

```
cars_unstacked
```

```
   year                          2019                          2020
  brand             VW         Mercedes           VW         Mercedes
  model   Golf Passat  A-class C-class Golf Passat  A-class C-class
feature
  price   25     31       41      32    26     32       42      33
   sold   68     72       68      75    72     73       69      74
```

Stacking by name or number

```
cars_unstacked.stack(level=[0, 1])
```

```
                model  A-class  C-class  Golf  Passat
feature year    brand
  price 2019 Mercedes     41.0     32.0   NaN     NaN
                   VW      NaN      NaN  25.0    31.0
        2020 Mercedes     42.0     33.0   NaN     NaN
                   VW      NaN      NaN  26.0    32.0
   sold 2019 Mercedes     68.0     75.0   NaN     NaN
                   VW      NaN      NaN  68.0    72.0
        2020 Mercedes     69.0     74.0   NaN     NaN
                   VW      NaN      NaN  72.0    73.0
```

```
cars_unstacked.stack(levels=['year', 'brand'])
```

```
                model  A-class  C-class  Golf  Passat
feature year    brand
  price 2019 Mercedes     41.0     32.0   NaN     NaN
                   VW      NaN      NaN  25.0    31.0
        2020 Mercedes     42.0     33.0   NaN     NaN
                   VW      NaN      NaN  26.0    32.0
   sold 2019 Mercedes     68.0     75.0   NaN     NaN
                   VW      NaN      NaN  68.0    72.0
        2020 Mercedes     69.0     74.0   NaN     NaN
                   VW      NaN      NaN  72.0    73.0
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Handling missing data

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Review

- Stack and unstack DataFrames:
  - All columns index levels
  - A row index level
  - Choose which levels to stack or unstack

Unstacking leads to missing values

Subgroups do not have the same set of labels

```
animals
```

```
                                jump  run  fly
class    order         name
Mammalia carnivora     dog        No  Yes   No
         Diprotodontia Kangaroo  Yes   No   No
Aves     hervibora     bird       No   No  Yes
```

Unstacking leads to missing values

Subgroups do not have the same set of labels

```
animals
```

```
                                jump  run  fly
class    order         name
   Mammalia carnivora     dog        No  Yes   No <--
         Diprotodontia Kangaroo  Yes   No   No
Aves     hervibora     bird       No   No  Yes
```

Unstacking leads to missing values

Subgroups do not have the same set of labels

```
animals.unstack(level='class')
```

```
                                jump            run           fly
        clas              Aves Mammalia Aves Mammalia Aves Mammalia
        order       name
  Diprotodontia Kangaroo  NaN      Yes  NaN       No  NaN       No
      carnivora      Dog  NaN       No  NaN      Yes  NaN       No
Charadriiformes   Avocet   No      NaN   No      NaN  Yes      NaN
```

Unstacking leads to missing values

Subgroups do not have the same set of labels

```
animals.unstack(level='class')
```

```
                                jump            run           fly
        clas              Aves Mammalia Aves Mammalia Aves Mammalia
        order       name
  Diprotodontia Kangaroo  NaN      Yes  NaN       No  NaN       No
  ----------------------------
      carnivora      Dog  NaN <--  No   NaN      Yes  NaN       No
  ----------------------------
Charadriiformes   Avocet   No      NaN   No      NaN  Yes      NaN
```

Handling NaN with unstack

```
animals.unstack(level='class', fill_value=    )
```

Handling NaN with unstack

```
animals.unstack(level='class', fill_value='No')
```

Handling NaN with unstack

```
animals.unstack(level='class', fill_value='No').sort_index(level=['order', 'name'], ascending=[True, False])
```

```
                                jump            run           fly
        clas              Aves Mammalia Aves Mammalia Aves Mammalia
        order       name
  Diprotodontia Kangaroo   No      Yes   No       No   No       No
      carnivora      Dog   No       No   No      Yes   No       No
Charadriiformes   Avocet   No       No   No       No  Yes       No
```

Stack and missing values

Combinations of index and column values missing from the original DataFrame

```
flowers
```

```
     petals Stigma
     number   size
rose     40    NaN
Lily      8    big
```

Stack and missing values

Combinations of index and column values missing from the original DataFrame

```
flowers.stack()
```

```
             Stigma  petals
rose number     NaN    40.0
Lily number     NaN     8.0
       size       5     NaN
```

Stack and missing values

Combinations of index and column values missing from the original DataFrame

```
flowers.stack(dropna=True)
```

```
             Stigma  petals
rose number     NaN    40.0
Lily number     NaN     8.0
       size       5     NaN
```

Stack and missing values

Combinations of index and column values missing from the original DataFrame

```
flowers.stack(dropna=False)
```

```
            Stigma  petals
rose number    NaN    40.0
         size    NaN     NaN <--
Lily number    NaN     8.0
       size      5     NaN
```

Handling NaN with stack

```
flowers.stack(dropna=False).fillna(0)
```

```
            Stigma  petals
rose number      0    40.0
       size      0       0
Lily number      0     8.0
       size      5       0
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Reshaping and combining data

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Reshaping and statistical functions

```
sales
```

```
                    office supply          Technology
         shop     online        onsite     online      onsite
country  year
Italy    2017      310           123         510        340
         2018      110           100         610        120
Spain    2017      229           200         300        240
         2018      120           220         190        210
```

Statistical functions

- Sum: `.sum()`
- Mean: `.mean()`
- Median: `.median()`
- Difference: `.diff()`

Stacking and stats

- Total amount of online and on-site sales by year in the two countries

```
sales.stack().sum(axis=1)
```

```
country  year   shop
Italy    2017  online    820
               onsite    463
         2018  online    720
               onsite    220
Spain    2017  online    529
               onsite    440
         2018  online    310
               onsite    430
```

Stacking and stats

- Total amount of online and on- site sales by year in the two countries

```
sales.stack().sum(axis=1).unstack()
```

```
         shop online onsite
country  year
 Italy   2017   820    463
         2018   720    220
 Spain   2017   529    440
         2018   310    430
```

Unstacking and stats

- Mean amount of product sales by year in both countries

```
sales.unstack(level=0).mean(axis=1)
```

```
year
2017    281.5
2018    210.0
```

Unstacking and stats

- Difference in the amount of sales between years

```
sales["office supply"].unstack(level='country')
```

Unstacking and stats

- Difference in the amount of sales between years

```
sales["office supply"].unstack(level='country').diff(axis=1, periods=2)
```

```
                          office supply
   shop          online           onsite
country   Italy  Spain   Italy   Spain
year
2017      NaN     NaN   -187.0   -29.0
2018      NaN     NaN    -10.0   100.0
```

Reshaping and grouping

- Total amount of different products by online or on-site regardless of the country

```
sales.stack().head(4)
```

```
                        office supply   Technology
country   year  shop
 Italy    2017 online        310           510
              onsite        123           340
          2018 online        110           610
              onsite        100           120
```

Reshaping and grouping

- Total amount of different products by online or on-site regardless of the country

```
sales.stack().groupby(level='shop').sum()
```

```
        office supply    Technology
shop
online        769             1610
onsite        643             910
```

Reshaping after grouping

- Median amount of products by year

```
sales.groupby(level='year').median()
```

```
        office supply          Technology
shop    online  onsite    online   onsite
year
2017     269.5  161.5      405.0   290.0
2018     115.0  160.0      400.0   165.0
```

Reshaping after grouping

- Median amount of products by year

```
sales.groupby(level=1).median().stack(level=[0, 1]).unstack(level='year')
```

```
                  year     2017     2018
                  shop
Technology       online   405.0    400.0
                 onsite   290.0    165.0
office supply    online   269.5    115.0
                 onsite   161.5    160.0
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Transforming a list-like column

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

List-like columns

	city	country	zip code
0	Los Angeles	USA	90001, 90004, 90008
1	Madrid	Spain	28001, 28004, 28005
2	Rabat	Morocco	10010, 10170

Transforming list-like columns

	city	country	zip code
0	Los Angeles	USA	90001, 90004, 90008
1	Madrid	Spain	28001, 28004, 28005
2	Rabat	Morocco	10010, 10170

	city	country	zip code
0	Los Angeles	USA	90001
1	Los Angeles	USA	90004
2	Los Angeles	USA	90008
3	Madrid	Spain	28001
4	Madrid	Spain	28004
5	Madrid	Spain	28005
6	Rabat	Morocco	10010
7	Rabat	Morocco	10170

The .explode() method

	city	country	zip code
0	Los Angeles	USA	90001, 90004, 90008
1	Madrid	Spain	28001, 28004, 28005
2	Rabat	Morocco	10010, 10170

	city	country	zip code
0	Los Angeles	USA	90001
1	Los Angeles	USA	90004
2	Los Angeles	USA	90008
3	Madrid	Spain	28001
4	Madrid	Spain	28004
5	Madrid	Spain	28005
6	Rabat	Morocco	10010
7	Rabat	Morocco	10170

```
df.explode()
```

Exploding a column

```
cities
```

```
          city  country                zip_code
0  Los Angeles      USA  [90001, 90004, 90008]
1       Madrid    Spain  [28001, 28004, 28005]
2        Rabat  Morocco         [10010, 10170]
```

Exploding a column

```
cities_explode = cities['zip_code'].explode()
cities_explode
```

```
0    90001
0    90004
0    90008
1    28001
1    28004
1    28005
2    10010
2    10170
```

Exploding a column

```
cities[['city', 'country']]
```

Exploding a column

```
cities[['city', 'country']].merge(cities_explode,                                    )
```

Exploding a column

```
cities[['city', 'country']].merge(cities_explode, left_index=True, right_index=True)
```

```
          city  country zip_code
0  Los Angeles      USA    90001
0  Los Angeles      USA    90004
0  Los Angeles      USA    90008
1       Madrid    Spain    28001
1       Madrid    Spain    28004
1       Madrid    Spain    28005
2        Rabat  Morocco    10010
2        Rabat  Morocco    10170
```

Exploding a column in the DataFrame

```
cities_explode = cities.explode('zip_code')
cities_explode
```

```
          city  country zip_code
0  Los Angeles      USA    90001
0  Los Angeles      USA    90004
0  Los Angeles      USA    90008
1       Madrid    Spain    28001
1       Madrid    Spain    28004
1       Madrid    Spain    28005
2        Rabat  Morocco    10010
2        Rabat  Morocco    10170
```

Exploding a column in the DataFrame

```
cities_explode.reset_index(drop=True, inplace=True)
```

```
          city  country zip_code
0  Los Angeles      USA    90001
1  Los Angeles      USA    90004
2  Los Angeles      USA    90008
3       Madrid    Spain    28001
4       Madrid    Spain    28004
5       Madrid    Spain    28005
6        Rabat  Morocco    10010
7        Rabat  Morocco    10170
```

Empty lists

```
cities_new
```

```
          city  country                zip_code
0  Los Angeles      USA  [90001, 90004, 90008]
1       Madrid    Spain                     []
2        Rabat  Morocco          [10010, 10170]
```

```
cities_new.explode('zip_code')
```

```
          city  country zip_code
0  Los Angeles      USA    90001
0  Los Angeles      USA    90004
0  Los Angeles      USA    90008
1       Madrid    Spain      NaN
2        Rabat  Morocco    10010
2        Rabat  Morocco    10170
```

Chaining operations

```
cities
```

```
          city  country             zip_code
0  Los Angeles      USA  90001, 90004, 90008
1       Madrid    Spain  28001, 28004, 28005
2        Rabat  Morocco         10010, 10170
```

Chaining operations

```
cities['zip_code'].str.split(',', expand=True)
```

```
       0      1      2
0  90001  90004  90008
1  28001  28004  28005
2  10010  10170   None
```

Chaining operations

```
cites.assign(zip_code=                                    )
```

Chaining operations

```
cites.assign(zip_code=cities['zip_code'].str.split(','))
```

Chaining operations

```
cites.assign(zip_code=cities['zip_code'].str.split(',')).explode('zip_code')
```

```
          city  country zip_code
0  Los Angeles      USA    90001
0  Los Angeles      USA    90004
0  Los Angeles      USA    90008
1       Madrid    Spain    28001
1       Madrid    Spain    28004
1       Madrid    Spain    28005
2        Rabat  Morocco    10010
2        Rabat  Morocco    10170
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Reading nested data into a DataFrame

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Review

- Reshape DataFrames and Series
- Explode lists contained in columns
- Split and concatenate strings

JSON format

- JavaScript Object Notation
- Data-interchange format
- Easy for humans to read and write
- Easy for machines to parse and generate

JSON format

```
my_writer
```

```
{
  "first" : "Mary",
  "last" : "Shelley",
  "country" : "England",
  "books" : 12
}
```

Nested JSON

```
writers
```

```
writers = [
            {
              "first": "Mary",
              "last": "Shelley",
              "books": {"title": "Frankenstein", "year": 1818}
            },
            {
              "first": "Ernest",
              "last": "Hemingway",
              "books": {"title": "The Old Man and the Sea", "year": 1951}
            }
          ]
```

Data normalization

```
from pandas import json_normalize
```

```
json_normalize(writers)
```

```
    first       last              books.title  books.year
0    Mary    Shelley             Frankenstein        1818
1  Ernest  Hemingway  The Old Man and the Sea        1951
```

Data normalization

```
writers_norm = json_normalize(writers, sep='_')
writers_norm
```

```
    first       last              books_title  books_year
0    Mary    Shelley             Frankenstein        1818
1  Ernest  Hemingway  The Old Man and the Sea        1951
```

Data normalization

```
pd.wide_to_long(writers_norm, stubnames=['books'], i=['first', 'last'], j='feature', sep='_', suffix='\w+')
```

```
                                          books
first  last      feature
Mary   Shelley   title               Frankenstein
                 year                        1818
Ernest Hemingway title    The Old Man and the Sea
                 year                        1951
```

Complex JSON

```
writers
```

```
[
    {'name': 'Mary',
     'last': 'Shelley',
     'books': [{'title': 'Frankestein', 'year': 1818},
               {'title': 'Mathilda ', 'year': 1819},
               {'title': 'The Last Man', 'year': 1826}]},
    {'name': 'Ernest',
     'last': 'Hemmingway',
     'books': [{'title': 'The Old Man and the Sea', 'year': 1951},
              {'title': 'The Sun Also Rises', 'year': 1927}]}
]
```

Complex JSON

```
json_normalize(writers)
```

```
     name        last                                                books
0    Mary     Shelley  [{'title': 'Frankestein', 'year': 1818}, {'tit...
1  Ernest  Hemmingway  [{'title': 'The Old Man and the Sea', 'year': ...
```

Record path

```
json_normalize(writers, record_path='books')
```

```
                     title  year
0              Frankestein  1818
1                Mathilda   1819
2             The Last Man  1826
3  The Old Man and the Sea  1951
4       The Sun Also Rises  1927
```

Metadata

```
json_normalize(writers, record_path='books', meta=['name', 'last'])
```

```
                    title  year    name        last
0              Frankestein  1818    Mary     Shelley
1                Mathilda   1819    Mary     Shelley
2             The Last Man  1826    Mary     Shelley
3  The Old Man and the Sea  1951  Ernest  Hemmingway
4       The Sun Also Rises  1927  Ernest  Hemmingway
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

Dealing with nested data columns

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

Review

- How to read nested JSON into DataFrame using `json_normalize()`.

Nested data in columns

```
writers = ["Mary Shelley", "Ernest Hemingway"]
books = ["{'title': 'Frankenstein', 'year': 1818}",
         "{'title': 'The Old Man and the Sea', 'year': 1951}"]
collection = pd.DataFrame(                                  )
```

Nested data in columns

```
writers = ["Mary Shelley", "Ernest Hemingway"]
books = ["{'title': 'Frankenstein', 'year': 1818}",
        "{'title': 'The Old Man and the Sea', 'year': 1951}"]
collection = pd.DataFrame(dict(                              ))
```

Nested data in columns

```
writers = ["Mary Shelley", "Ernest Hemingway"]
books = ['{"title": "Frankenstein", "year": "1818"}',
         '{"title": "The Old Man and the Sea", "year":"1951"}']
collection = pd.DataFrame(dict(writers=writers, books=books))
collection
```

```
             writers                                                books
0       Mary Shelley              {'title': 'Frankenstein', 'year': 1818}
1  Ernest Hemingway  {'title': 'The Old Man and the Sea', 'year': 1951}
```

Converting nested data

```
import json
books = collection['books']
```

Converting nested data

```
import json
books = collection['books'].apply(          )
```

Converting nested data

```
import json
books = collection['books'].apply(json.loads)
```

Converting nested data

```
import json
books = collection['books'].apply(json.loads).apply(pd.Series)
books
```

```
                      title  year
0              Frankenstein  1818
1  The Old Man and the Sea  1951
```

Concatenate back

```
collection = collection.drop(columns='books')
pd.concat([collection, books], axis=1)
```

```
            writers                     title  year
0      Mary Shelley              Frankenstein  1818
1  Ernest Hemingway  The Old Man and the Sea  1951
```

Dumping nested data

```
import json
books = collection['books'].apply(json.loads)
```

Dumping nested data

```
import json
books = collection['books'].apply(json.loads).to_list()
books_dump = json.dumps(books)
new_books = pd.read_json(books_dump)
new_books
```

```
                     title  year
0             Frankenstein  1818
1  The Old Man and the Sea  1951
```

Dumping nested data

```
pd.concat([collection['writers'], new_books], axis=1)
```

```
            writers                     title  year
0      Mary Shelley              Frankenstein  1818
1  Ernest Hemingway  The Old Man and the Sea  1951
```

Let's practice!

RESHAPING DATA WITH PANDAS

datacamp

The final reshape

RESHAPING DATA WITH PANDAS

Maria Eugenia Inzaugarat
Data Scientist

datacamp

	Message
0	CON
1	GRA
2	TU
3	LA
4	TIONS

Message					
	CON	GRA	TU	LA	TIONS

Chapter 1

	Chapter 1	Chapter 2	Chapter 3	Chapter 4
0	Long & Wide	melt	stack	Reshape &Combining
1	pivot	wide_to_long	unstack	List-like col
2	pivot_table	string col	np.nan	Nested data

Concept of long and wide formats

Use `.pivot()` method - columns as unique variables, index as individual observations

Create pivot tables

Learn the difference between `.pivot()` and `.pivot_table()`

Chapter 2

	Chapter 1	Chapter 2	Chapter 3	Chapter 4
0	Long & Wide	melt	stack	Reshape &Combining
1	pivot	wide_to_long	unstack	List-like col
2	pivot_table	string col	np.nan	Nested data

From a wide to a long format using:

- the `.melt()` method
- the `wide_to_long()` function

Splitting or concatenating string columns

Chapter 3

	Chapter 1	Chapter 2	Chapter 3	Chapter 4
0	Long & Wide	melt	stack	Reshape & Combining
1	pivot	wide_to_long	unstack	List-like col
2	pivot_table	string col	np.nan	Nested data

Multi-level index

Use `.stack()` and `.unstack()`

Handle generated missing data

Chapter 4

	Chapter 1	Chapter 2	Chapter 3	Chapter 4
0	Long & Wide	melt	stack	Reshape &Combining
1	pivot	wide_to_long	unstack	List-like col
2	pivot_table	string col	np.nan	Nested data

Combine reshaping and grouping processes

List-like column transformation

Nested data in columns

Thank you!

RESHAPING DATA WITH PANDAS