Data Wrangling

TUAN NGUYEN

Outline

- Understanding data wrangling
- Cleaning data
- Reshaping data
- Handling duplicate, missing, or invalid data

Data wrangling

- Data wrangling (data manipulation): we are taking our input data from its original state and putting it in a format where we can perform meaningful analysis on it.
- There are three common tasks involved in the data wrangling process:
  - Data cleaning
  - Data transformation
  - Data enrichment

Data cleaning

Some essential data cleaning tasks to master include the following:

- Renaming
- Sorting and reordering
- Data type conversions
- Handling duplicate data
- Addressing missing or invalid data
- Filtering to the desired subset of data

Data transformation

- In data transformation, we focus on changing our data's structure to facilitate our downstream analyses.
- This usually involves changing which data goes along the rows and which goes down the column.
- Most data we will find is either wide format or long format.

Wide vs Long format

Wide format

variables

observations

	date	TMAX	TMIN	TOBS
0	2018-10-01	21.1	8.9	13.9
1	2018-10-02	23.9	13.9	17.2
2	2018-10-03	25.0	15.6	16.1
3	2018-10-04	22.8	11.7	11.7
4	2018-10-05	23.3	11.7	18.9
5	2018-10-06	20.0	13.3	16.1

Long format

variable names

variable values

repeated values for **date** column

	date	datatype	value
0	2018-10-01	TMAX	21.1
1	2018-10-01	TMIN	8.9
2	2018-10-01	TOBS	13.9
3	2018-10-02	TMAX	23.9
4	2018-10-02	TMIN	13.9
5	2018-10-02	TOBS	17.2

Wide data format

- Represent measurements of variables with their own columns, and each row represents an observation of those variables.
- This makes it easy for us to compare variables across observations, get summary statistics, perform operations, and present our data.

	date	TMAX	TMIN	TOBS
0	2018-10-01	21.1	8.9	13.9
1	2018-10-02	23.9	13.9	17.2
2	2018-10-03	25.0	15.6	16.1
3	2018-10-04	22.8	11.7	11.7
4	2018-10-05	23.3	11.7	18.9
5	2018-10-06	20.0	13.3	16.1

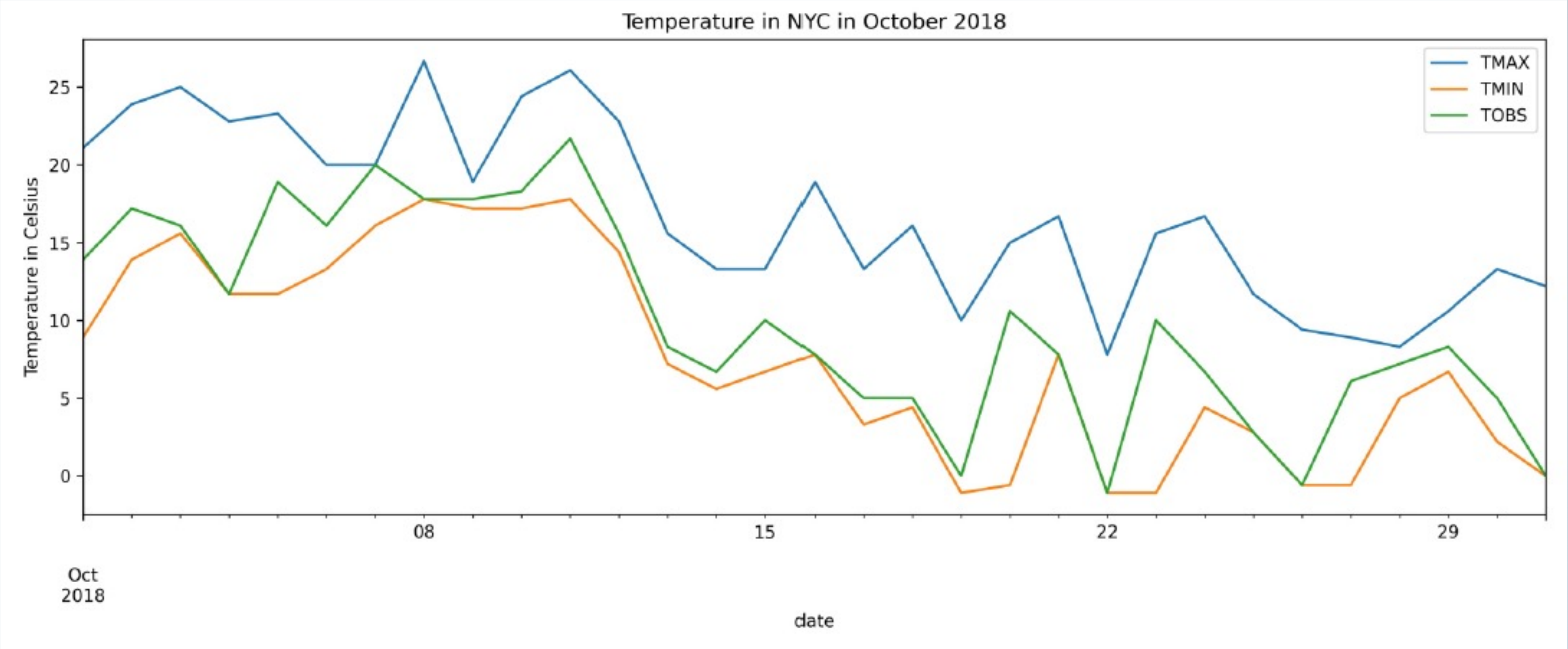
Wide data format

```
wide_df.describe(include='all', datetime_is_numeric=True)
```

	date	TMAX	TMIN	TOBS
count	31	31.000000	31.000000	31.000000
mean	2018-10-16 00:00:00	16.829032	7.561290	10.022581
min	2018-10-01 00:00:00	7.800000	-1.100000	-1.100000
25%	2018-10-08 12:00:00	12.750000	2.500000	5.550000
50%	2018-10-16 00:00:00	16.100000	6.700000	8.300000
75%	2018-10-23 12:00:00	21.950000	13.600000	16.100000
max	2018-10-31 00:00:00	26.700000	17.800000	21.700000
std	NaN	5.714962	6.513252	6.596550

Wide data format

```
wide_df.plot(
    x='date', y=['TMAX', 'TMIN', 'TOBS'], figsize=(15, 5),
    title='Temperature in NYC in October 2018'
).set_ylabel('Temperature in Celsius')
plt.show()
```



Long data format

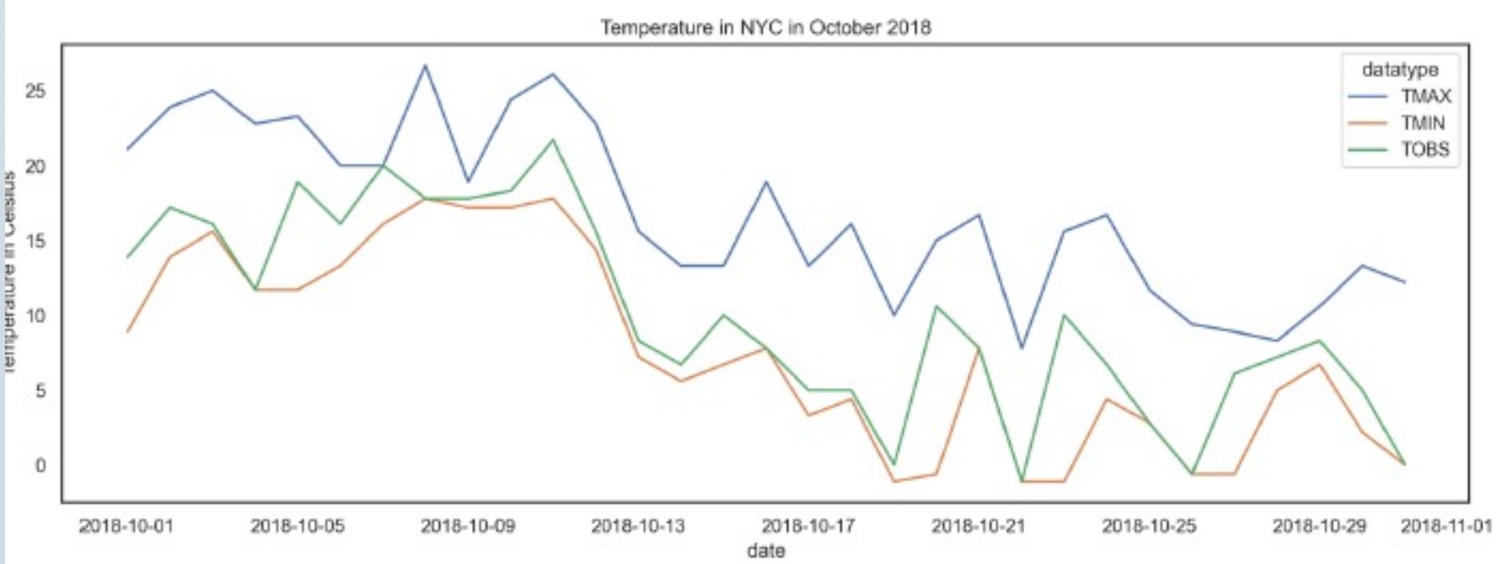
- Long format data will have a row for each observation of a variable.
- If we have three variables being measured daily, we will have three rows for each day we record observations.

	date	datatype	value
0	2018-10-01	TMAX	21.1
1	2018-10-01	TMIN	8.9
2	2018-10-01	TOBS	13.9
3	2018-10-02	TMAX	23.9
4	2018-10-02	TMIN	13.9
5	2018-10-02	TOBS	17.2

Long data format

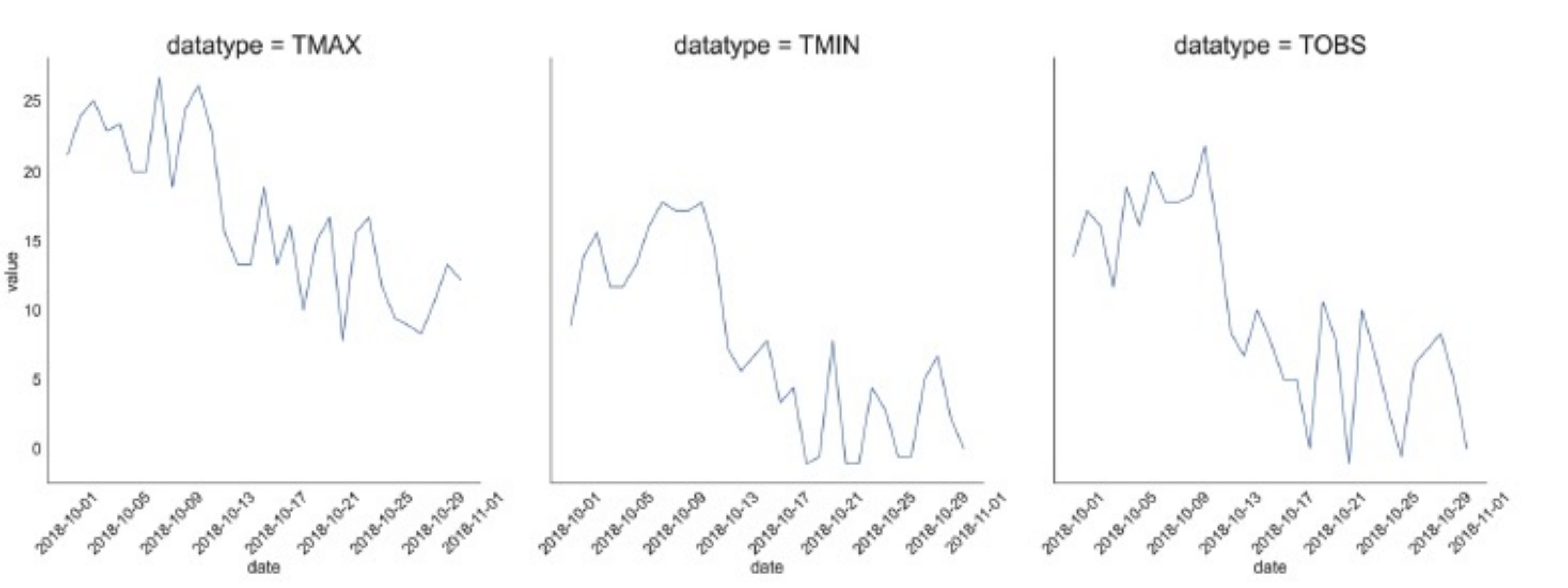
It makes it easy to create visualizations where our plotting library can color lines by the name of the variable, size the points by the values of a certain variable, and erform splits for faceting

```
import seaborn as sns

sns.set(rc={'figure.figsize': (15, 5)}, style='white')

ax = sns.lineplot(
    data=long_df, x='date', y='value', hue='datatype'
)
```



Long data format

```
sns.set(
    rc={'figure.figsize': (20, 10)},
    style='white', font_scale=2
)
g = sns.FacetGrid(long_df, col='datatype', height=10)
g = g.map(plt.plot, 'date', 'value')
g.set_titles(size=25)
g.set_xticklabels(rotation=45)
plt.show()
```



Data enrichment

- Data enrichment improves the quality of the data by adding to it in one way or another. This process becomes very important in modeling and in machine learning, where it forms part of the feature engineering process.
- The following are ways to enhance our data using the original data:
  - Adding new columns: Using functions on the data from existing columns to create new values.
  - Binning: Turning continuous data or discrete data with many distinct values into buckets, which makes the column discrete while letting us control the number of possible values in the column.
  - Aggregating: Rolling up the data and summarizing it.
  - Resampling: Aggregating time series data at specific intervals.

DATA CLEANING

Temperature data

```
import pandas as pd
df = pd.read_csv('data/nyc_temperatures.csv')
df.head()
```

	date	datatype	station	attributes	value
0	2018-10-01T00:00:00	TAVG	GHCND:USW00014732	H,,S,	21.2
1	2018-10-01T00:00:00	TMAX	GHCND:USW00014732	,,W,2400	25.6
2	2018-10-01T00:00:00	TMIN	GHCND:USW00014732	,,W,2400	18.3
3	2018-10-02T00:00:00	TAVG	GHCND:USW00014732	H,,S,	22.7
4	2018-10-02T00:00:00	TMAX	GHCND:USW00014732	,,W,2400	26.1

Renaming columns

```
>>> df.columns
Index(['date', 'datatype', 'station', 'attributes', 'value'],
      dtype='object')
```

```
df.rename(
    columns={'value': 'temp_C', 'attributes': 'flags'},
    inplace=True
)
```

```
>>> df.columns
Index(['date', 'datatype', 'station', 'flags', 'temp_C'],
      dtype='object')
```

Type conversion

```
>>> df.dtypes
date              object
datatype          object
station           object
flags             object
temp_C           float64
dtype: object
```

```
>>> df.loc[:,'date'] = pd.to_datetime(df.date)
>>> df.dtypes
date        datetime64[ns]
datatype            object
station             object
flags               object
temp_C             float64
dtype: object
```

Type conversion

```
df = pd.read_csv('data/nyc_temperatures.csv').rename(
    columns={'value': 'temp_C', 'attributes': 'flags'}
)

new_df = df.assign(
    date=pd.to_datetime(df.date),
    temp_F=(df.temp_C * 9/5) + 32
)
```

```
>>> new_df.dtypes
date        datetime64[ns]
datatype            object
station             object
flags               object
temp_C             float64
temp_F             float64
dtype: object
```

	date	datatype	station	flags	temp_C	temp_F
0	2018-10-01	TAVG	GHCND:USW00014732	H,,S,	21.2	70.16
1	2018-10-01	TMAX	GHCND:USW00014732	,,W,2400	25.6	78.08
2	2018-10-01	TMIN	GHCND:USW00014732	,,W,2400	18.3	64.94

Type conversion

```
df = df.assign(
    date=lambda x: pd.to_datetime(x.date),
    temp_C_whole=lambda x: x.temp_C.astype('int'),
    temp_F=lambda x: (x.temp_C * 9/5) + 32,
    temp_F_whole=lambda x: x.temp_F.astype('int')
)
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_whole
0	2018-10-01	TAVG	GHCND:USW00014732	H,,S,	21.2	21	70.16	70
1	2018-10-01	TMAX	GHCND:USW00014732	,,W,2400	25.6	25	78.08	78
2	2018-10-01	TMIN	GHCND:USW00014732	,,W,2400	18.3	18	64.94	64

Sort data

```
df[df.datatype == 'TMAX']\
    .sort_values(by='temp_C', ascending=False).head(10)
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_whole
19	2018-10-07	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	82
28	2018-10-10	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	82
31	2018-10-11	TMAX	GHCND:USW00014732	,,W,2400	26.7	26	80.06	80
10	2018-10-04	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78
4	2018-10-02	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78

Sort data

```
df[df.datatype == 'TMAX'].sort_values(
    by=['temp_C', 'date'], ascending=[False, True]
).head(10)
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_whole
19	2018-10-07	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	82
28	2018-10-10	TMAX	GHCND:USW00014732	,,W,2400	27.8	27	82.04	82
31	2018-10-11	TMAX	GHCND:USW00014732	,,W,2400	26.7	26	80.06	80
4	2018-10-02	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78
10	2018-10-04	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78

Sort index

```
>>> df.sample(5, random_state=0).index
Int64Index([2, 30, 55, 16, 13], dtype='int64')
>>> df.sample(5, random_state=0).sort_index().index
Int64Index([2, 13, 16, 30, 55], dtype='int64')
```

```
df.sort_index(axis=1).head()
```

	datatype	date	flags	station	temp_C	temp_C_whole	temp_F	temp_F_whole
0	TAVG	2018-10-01	H,,S,	GHCND:USW00014732	21.2	21	70.16	70
1	TMAX	2018-10-01	,,W,2400	GHCND:USW00014732	25.6	25	78.08	78
2	TMIN	2018-10-01	,,W,2400	GHCND:USW00014732	18.3	18	64.94	64
3	TAVG	2018-10-02	H,,S,	GHCND:USW00014732	22.7	22	72.86	72
4	TMAX	2018-10-02	,,W,2400	GHCND:USW00014732	26.1	26	78.98	78

Set index

```
df.set_index('date', inplace=True)
df.head()
```

	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_whole
date							
2018-10-01	TAVG	GHCND:USW00014732	H,,S,	21.2	21	70.16	70
2018-10-01	TMAX	GHCND:USW00014732	,,W,2400	25.6	25	78.08	78
2018-10-01	TMIN	GHCND:USW00014732	,,W,2400	18.3	18	64.94	64
2018-10-02	TAVG	GHCND:USW00014732	H,,S,	22.7	22	72.86	72
2018-10-02	TMAX	GHCND:USW00014732	,,W,2400	26.1	26	78.98	78

Reset index

```
df['2018-10-11':'2018-10-12'].reset_index()
```

	date	datatype	station	flags	temp_C	temp_C_whole	temp_F	temp_F_whole
0	2018-10-11	TAVG	GHCND:USW00014732	H,,S,	23.4	23	74.12	74
1	2018-10-11	TMAX	GHCND:USW00014732	,,W,2400	26.7	26	80.06	80
2	2018-10-11	TMIN	GHCND:USW00014732	,,W,2400	21.7	21	71.06	71
3	2018-10-12	TAVG	GHCND:USW00014732	H,,S,	18.3	18	64.94	64
4	2018-10-12	TMAX	GHCND:USW00014732	,,W,2400	22.2	22	71.96	71
5	2018-10-12	TMIN	GHCND:USW00014732	,,W,2400	12.2	12	53.96	53

```
sp = pd.read_csv(
    'data/sp500.csv', index_col='date', parse_dates=True
).drop(columns=['adj_close']) # not using this column
```

Exercise

We want to look at data for the Facebook, Apple, Amazon, Netflix, and Google (FAANG) stocks. Combine them into a single file and store the dataframe of the FAANG data as faang for the rest of the exercises:

a) Read in the aapl.csv, amzn.csv, fb.csv, goog.csv, and nflx.csv files.
b) Add a column to each dataframe, called ticker, indicating the ticker symbol it is for (Apple's is AAPL, for example); this is how you look up a stock. In this case, the filenames happen to be the ticker symbols.
c) Append them together into a single dataframe. (concat)
d) Save the result in a CSV file called faang.csv. (to_csv)
e) With faang, use type conversion to cast the values of the date column into datetimes and the volume column into integers. Then, sort by date and ticker.
f) Find the seven rows in faang with the lowest value for volume

RESHAPING DATA

Pivoting dataframes

- We pivot our data to go from long format to wide format. The pivot() method performs this restructuring of our DataFrame object.
- To pivot, we need to tell pandas which column currently holds the values (with the values argument) and the column that contains what will become the column names in wide format (the columns argument).
- We can provide a new index (the index argument).

Pivoting dataframes

	datatype	date	temp_C	temp_F
0	TMAX	2018-10-01	21.1	69.98
1	TMIN	2018-10-01	8.9	48.02
2	TOBS	2018-10-01	13.9	57.02
3	TMAX	2018-10-02	23.9	75.02
4	TMIN	2018-10-02	13.9	57.02

```
pivoted_df = long_df.pivot(
    index='date', columns='datatype', values='temp_C'
)
pivoted_df.head()
```

datatype	TMAX	TMIN	TOBS
date			
2018-10-01	21.1	8.9	13.9
2018-10-02	23.9	13.9	17.2
2018-10-03	25.0	15.6	16.1
2018-10-04	22.8	11.7	11.7
2018-10-05	23.3	11.7	18.9

Pivoting dataframes

```
pivoted_df = long_df.pivot(
    index='date', columns='datatype',
    values=['temp_C', 'temp_F']
)
pivoted_df.head()
```

	temp_C			temp_F		
datatype	**TMAX**	**TMIN**	**TOBS**	**TMAX**	**TMIN**	**TOBS**
date						
2018-10-01	21.1	8.9	13.9	69.98	48.02	57.02
2018-10-02	23.9	13.9	17.2	75.02	57.02	62.96
2018-10-03	25.0	15.6	16.1	77.00	60.08	60.98
2018-10-04	22.8	11.7	11.7	73.04	53.06	53.06
2018-10-05	23.3	11.7	18.9	73.94	53.06	66.02

```
>>> pivoted_df['temp_F']['TMIN'].head()
date
2018-10-01    48.02
2018-10-02    57.02
2018-10-03    60.08
2018-10-04    53.06
2018-10-05    53.06
Name: TMIN, dtype: float64
```

MultiIndex

```
>>> multi_index_df = long_df.set_index(['date', 'datatype'])

>>> multi_index_df.head().index
MultiIndex([('2018-10-01', 'TMAX'),
            ('2018-10-01', 'TMIN'),
            ('2018-10-01', 'TOBS'),
            ('2018-10-02', 'TMAX'),
            ('2018-10-02', 'TMIN')],
           names=['date', 'datatype'])
```

		temp_C	temp_F
date	**datatype**		
2018-10-01	**TMAX**	21.1	69.98
	TMIN	8.9	48.02
	TOBS	13.9	57.02
2018-10-02	**TMAX**	23.9	75.02
	TMIN	13.9	57.02

Exercise

The European Centre for Disease Prevention and Control (ECDC) provides an open dataset on COVID-19 cases called daily number of new reported cases of COVID-19 by country worldwide. This dataset is updated daily, but we will use a snapshot that contains data from January 1, 2020 through September 18, 2020. Clean and pivot the data so that it is in wide format:

a) Read in the covid19_cases.csv file.

b) Create a date column using the data in the dateRep column and the pd.to_datetime() function.

c) Set the date column as the index and sort the index.

d) Replace all occurrences of United_States_of_America and United_ Kingdom with USA and UK, respectively. Hint: the replace() method can be run on the dataframe as a whole.

e) Using the countriesAndTerritories column, filter the cleaned COVID-19 cases data down to Argentina, Brazil, China, Colombia, India, Italy, Mexico, Peru, Russia, Spain, Turkey, the UK, and the USA.

f) Pivot the data so that the index contains the dates, the columns contain the country names, and the values are the case counts (the cases column). Be sure to fill in NaN values with 0.

Melting dataframe

- To go from wide format to long format, we need to melt the data. Melting undoes a pivot.
- We can use the melt() method for flexible reshaping—allowing us to turn this into long format. Melting requires that we specify the following:
  - which column(s) uniquely identify a row in the wide format data with the id_vars argument
  - which column(s) contain(s) the variable(s) with the value_vars argument
  - we can also specify how to name the column containing the variable names in the long format data (var_name) and the name for the column containing their values (value_name)

Melting dataframe

```
melted_df = wide_df.melt(
    id_vars='date', value_vars=['TMAX', 'TMIN', 'TOBS'],
    value_name='temp_C', var_name='measurement'
)
melted_df.head()
```

	date	TMAX	TMIN	TOBS
0	2018-10-01	21.1	8.9	13.9
1	2018-10-02	23.9	13.9	17.2
2	2018-10-03	25.0	15.6	16.1
3	2018-10-04	22.8	11.7	11.7
4	2018-10-05	23.3	11.7	18.9

	date	measurement	temp_C
0	2018-10-01	TMAX	21.1
1	2018-10-02	TMAX	23.9
2	2018-10-03	TMAX	25.0
3	2018-10-04	TMAX	22.8
4	2018-10-05	TMAX	23.3

HANDLING DUPLICATE, MISSING, INVALID DATA

Problematic data

	date	station	PRCP	SNOW	SNWD	TMAX	TMIN	TOBS	WESF	inclement_weather
0	2018-01-01T00:00:00	?	0.0	0.0	-inf	5505.0	-40.0	NaN	NaN	NaN
1	2018-01-01T00:00:00	?	0.0	0.0	-inf	5505.0	-40.0	NaN	NaN	NaN
2	2018-01-01T00:00:00	?	0.0	0.0	-inf	5505.0	-40.0	NaN	NaN	NaN
3	2018-01-02T00:00:00	GHCND:USC00280907	0.0	0.0	-inf	-8.3	-16.1	-12.2	NaN	False
4	2018-01-03T00:00:00	GHCND:USC00280907	0.0	0.0	-inf	-4.4	-13.9	-13.3	NaN	False

Problematic data

	PRCP	SNOW	SNWD	TMAX	TMIN	TOBS	WESF
count	765.000000	577.000000	577.0	765.000000	765.000000	398.000000	11.000000
mean	5.360392	4.202773	NaN	2649.175294	-15.914379	8.632161	16.290909
std	10.002138	25.086077	NaN	2744.156281	24.242849	9.815054	9.489832
min	0.000000	0.000000	-inf	-11.700000	-40.000000	-16.100000	1.800000
25%	0.000000	0.000000	NaN	13.300000	-40.000000	0.150000	8.600000
50%	0.000000	0.000000	NaN	32.800000	-11.100000	8.300000	19.300000
75%	5.800000	0.000000	NaN	5505.000000	6.700000	18.300000	24.900000
max	61.700000	229.000000	inf	5505.000000	23.900000	26.100000	28.700000

Problematic data

```
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 765 entries, 0 to 764
Data columns (total 10 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   date                  765 non-null    object
 1   station               765 non-null    object
 2   PRCP                  765 non-null    float64
 3   SNOW                  577 non-null    float64
 4   SNWD                  577 non-null    float64
 5   TMAX                  765 non-null    float64
```

Problematic data

```
>>> contain_nulls = df[
...     df.SNOW.isna() | df.SNWD.isna() | df.TOBS.isna()
...     | df.WESF.isna() | df.inclement_weather.isna()
... ]
>>> contain_nulls.shape[0]
765
>>> contain_nulls.head(10)
```

	date	station	PRCP	SNOW	SNWD	TMAX	TMIN	TOBS	WESF	inclement_weather
0	2018-01-01T00:00:00	?	0.0	0.0	-inf	5505.0	-40.0	NaN	NaN	NaN
1	2018-01-01T00:00:00	?	0.0	0.0	-inf	5505.0	-40.0	NaN	NaN	NaN
2	2018-01-01T00:00:00	?	0.0	0.0	-inf	5505.0	-40.0	NaN	NaN	NaN
3	2018-01-02T00:00:00	GHCND:USC00280907	0.0	0.0	-inf	-8.3	-16.1	-12.2	NaN	False
4	2018-01-03T00:00:00	GHCND:USC00280907	0.0	0.0	-inf	-4.4	-13.9	-13.3	NaN	False

Problematic data

	date	station	inclement_weather
count	765	765	408
unique	324	2	2
top	2018-07-05T00:00:00	GHCND:USC00280907	False
freq	8	398	384

```
>>> df[df.duplicated()].shape[0]
284
```

```
>>> df[df.duplicated(keep=False)].shape[0]
482
```

	date	station	PRCP	SNOW	SNWD
1	2018-01-01T00:00:00	?	0.0	0.0	-inf
2	2018-01-01T00:00:00	?	0.0	0.0	-inf
5	2018-01-03T00:00:00	GHCND:USC00280907	0.0	0.0	-inf
6	2018-01-03T00:00:00	GHCND:USC00280907	0.0	0.0	-inf