This notebook is an exercise in the Data Cleaning course. You can reference the tutorial at this link.

In this exercise, you'll apply what you learned in the **Parsing dates** tutorial.

Setup

The questions below will give you feedback on your work. Run the following cell to set up the feedback system.

In [ ]:
```
from learntools.core import binder
binder.bind(globals())
from learntools.data_cleaning.ex3 import *
print("Setup Complete")
```

Get our environment set up

The first thing we'll need to do is load in the libraries and dataset we'll be using. We'll be working with a dataset containing information on earthquakes that occured between 1965 and 2016.

In [ ]:
```
# modules we'll use
import pandas as pd
import numpy as np
import seaborn as sns
import datetime

# read in our data
earthquakes = pd.read_csv("../input/earthquake-database/database.csv")
```

```
# set seed for reproducibility
np.random.seed(0)
```

1) Check the data type of our date column

You'll be working with the "Date" column from the `earthquakes` dataframe. Investigate this column now: does it look like it contains dates? What is the dtype of the column?

In [ ]:
```
print(earthquakes['Date'].head())
```

In [ ]:
```
# TODO: Your code here!

earthquakes['Date'].dtype
```

In [ ]:
```

```

Once you have answered the question above, run the code cell below to get credit for your work.

In [ ]:
```
# Check your answer (Run this code cell to receive credit!)
q1.check()
```

In [ ]:
```
# Line below will give you a hint
#q1.hint()
```

2) Convert our date columns to datetime

Most of the entries in the "Date" column follow the same format: "month/day/four-digit year". However, the entry at index 3378 follows a completely different pattern. Run the code cell below to see this.

In [ ]:
```
earthquakes[3378:3383]
```

In [ ]:
```
earthquakes["date_parsed"]=pd.to_datetime(earthquakes["Date"], format=
"%m/%d/%Y", errors="coerce")
invalid_date_index=earthquakes["date_parsed"][earthquakes["date_parsed"
].isnull()==True].index.tolist()
```

In [ ]:
```
fixed_date=[]
for index in invalid_date_index:
    sliced=earthquakes.loc[index,["Date"]].values[0][:10]
    fixed_date.append((index, sliced))
```

In [ ]:
```
for fixed in fixed_date:
    earthquakes.loc[fixed[0], "date_parsed"]=pd.to_datetime(fixed[1], f
ormat="%Y-%m-%d")
```

In [ ]:
```
# Your turn! Create a new column, date_parsed, in the earthquakes
# dataset that has correctly parsed dates in it. (Don't forget to
# double-check that the dtype is correct!)

# There are invalid date format in the data.
# My goal is to locate those data and then decide how to deal with them
individually.
# Step1: get the index of invalid date
earthquakes["date_parsed"]=pd.to_datetime(earthquakes["Date"], format=
"%m/%d/%Y", errors="coerce")
invalid_date_index=earthquakes["date_parsed"][earthquakes["date_parsed"
].isnull()==True].index.tolist()
# Step2: print out invalid date
for index in invalid_date_index:
    print("index {} has date {}".format(index, earthquakes.loc[index,[
"Date"]].values))
# Step3: fix invalid date one by one
## As we see, we can still parse the date we want by slicing the origin
al date string.
fixed_date=[]
for index in invalid_date_index:
    sliced=earthquakes.loc[index,["Date"]].values[0][:10]
    fixed_date.append((index, sliced))
## then put the correct date format back to our dataframe
```

```
for fixed in fixed_date:
    earthquakes.loc[fixed[0], "date_parsed"]=pd.to_datetime(fixed[1], f
ormat="%Y-%m-%d")
## check fixed values in column "date_parsed"
earthquakes.iloc[[3378, 7512, 20650]]
```

This does appear to be an issue with data entry: ideally, all entries in the column have the same format. We can get an idea of how widespread this issue is by checking the length of each entry in the "Date" column.

In [ ]:
```
date_lengths = earthquakes.Date.str.len()
date_lengths.value_counts()
```

Looks like there are two more rows that has a date in a different format. Run the code cell below to obtain the indices corresponding to those rows and print the data.

In [ ]:
```
indices = np.where([date_lengths == 24])[1]
print('Indices with corrupted data:', indices)
earthquakes.loc[indices]
```

Given all of this information, it's your turn to create a new column "date_parsed" in the `earthquakes` dataset that has correctly parsed dates in it.

Note: When completing this problem, you are allowed to (but are not required to) amend the entries in the "Date" and "Time" columns. Do not remove any rows from the dataset.

In [ ]:
```
# TODO: Your code here

# Check your answer
q2.check()
```

In [ ]:
```
# Lines below will give you a hint or solution code
#q2.hint()
#q2.solution()
```

3) Select the day of the month

Create a Pandas Series `day_of_month_earthquakes` containing the day of the month from the "date_parsed" column.

In [ ]:
```
# try to get the day of the month from the date column
# Your turn! get the day of the month from the date_parsed column
day_of_month_earthquakes = earthquakes['date_parsed'].dt.day
day_of_month_earthquakes.head()

# Check your answer
q3.check()
```

In [ ]:
```
# Lines below will give you a hint or solution code
#q3.hint()
#q3.solution()
```

4) Plot the day of the month to check the date parsing

Plot the days of the month from your earthquake dataset.

In [ ]:
```
# TODO: Your code here!
day_of_month_earthquakes = day_of_month_earthquakes.dropna()

# plot the day of the month
sns.distplot(day_of_month_earthquakes, kde=False, bins=31)
```

Does the graph make sense to you?

In [ ]:
```
# Check your answer (Run this code cell to receive credit!)
q4.check()
```

In [ ]:
```
# Line below will give you a hint
#q4.hint()
```

(Optional) Bonus Challenge

For an extra challenge, you'll work with a Smithsonian dataset that documents Earth's volcanoes and their eruptive history over the past 10,000 years

Run the next code cell to load the data.

In [ ]:
```
volcanos = pd.read_csv("../input/volcanic-eruptions/database.csv")
```

Try parsing the column "Last Known Eruption" from the `volcanos` dataframe. This column contains a mixture of text ("Unknown") and years both before the common era (BCE, also known as BC) and in the common era (CE, also known as AD).

In [ ]:
```
volcanos['Last Known Eruption'].sample(5)
```

(Optional) More practice

If you're interested in graphing time series, check out this tutorial.

You can also look into passing columns that you know have dates in them the `parse_dates` argument in `read_csv`. (The documention is here.) Do note that this method can be very slow, but depending on your needs it may sometimes be handy to use.

Keep going

In the next lesson, learn how to **work with character encodings**.

Have questions or comments? Visit the Learn Discussion forum to chat with other Learners.