Lab Assignment 9: Data Management Using `pandas`, Part 2

DS 6001: Practice and Application of Data Science

H. Diana McSpadden (hdm5s)

Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

Problem 0

Import the following libraries:

In [ ]:

```
import numpy as np
import pandas as pd
```

Problem 1

In the first part of this lab, the goal is to merge data from the United Nations World Health Organization (https://www.who.int/who-un/en/) with data from the Varieties of Democracy Project (https://www.v-dem.net/en/). The UN-WHO studies health outcomes in a cross-national context, and V-Dem studies the quality of democracy as it changes across countries and over time. We would want to merge these two datasets together if we wanted to study **whether democratic quality can predict health outcomes.**

The UN data contains cross-national time series data from the United Nations and World Health Organization, and includes three features:

- The number of physicians per 1000 people
- The percent of the population that is malnourished
- Health expenditure per capita

The VDem data comes from the Varieties of Democracy project, which aims to measure the quality of democracy and the amount of corruption in different countries over time (https://www.v-dem.net/en/data/data-version-8/). This data file contains

- indices regarding a country's democractic quality,
- level of civil liberites,
- level of corruption.
- a binary indicator that separates countries into democratic and nondemocratic states,
- a categorizaton of the corruption scale.

The URLs for the two datasets are:

In [ ]:
```
undata_url = "https://github.com/jkropko/DS-6001/raw/master/localdata/UNdata.csv"
VDem_url = "https://github.com/jkropko/DS-6001/raw/master/localdata/vdem.csv"
```

Part a

Load both CSV files. Make sure to check whether there are rows that should not be included in the dataframe, and whether there are missing codes that should be replaced with `NaN`. Fix these problems at the data loading stage, if you can. (Don't worry about column names or category labels yet.) Also, the UN data covers the years 1960-2014, and the VDem data covers the years 1960-2015. To make the timeframe match up, delete rows in the VDem data from 2015. (1 point)

Answer Part a

I will start with the **UN Data**

In [ ]:
```
pd.set_option('display.max_columns', 500)
pd.set_option('display.max_rows', 10)
```

In [ ]:
```
df_undata = pd.read_csv(undata_url, na_values="..", skipfooter = 5)
print(df_undata.shape)
df_undata
```

```
C:\Users\dianam\AppData\Local\Temp\ipykernel_2904\1335199655.py:1: ParserWarning: 
Falling back to the 'python' engine because the 'c' engine does not support skipfo
oter; you can avoid this warning by specifying engine='python'.
  df_undata = pd.read_csv(undata_url, na_values="..", skipfooter = 5)
(774, 60)
```

Out[ ]:

	Series Name	Series Code	Country Name	Country Code	1960 [YR1960]	1961 [YR1961]	1962 [YR1962]	1963 [YR1963]
0	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	Afghanistan	AFG	0.034844	NaN	NaN	NaN
1	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	Albania	ALB	0.276291	NaN	NaN	NaN
2	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	Algeria	DZA	0.173148	NaN	NaN	NaN
3	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	American Samoa	ASM	NaN	NaN	NaN	NaN
4	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	Andorra	ADO	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
769	Health expenditure per capita (current US$)	SH.XPD.PCAP	West Bank and Gaza	WBG	NaN	NaN	NaN	NaN
770	Health expenditure per capita (current US$)	SH.XPD.PCAP	World	WLD	NaN	NaN	NaN	NaN
771	Health expenditure per capita (current US$)	SH.XPD.PCAP	Yemen, Rep.	YEM	NaN	NaN	NaN	NaN
772	Health expenditure per capita (current US$)	SH.XPD.PCAP	Zambia	ZMB	NaN	NaN	NaN	NaN
773	Health expenditure per capita (current US$)	SH.XPD.PCAP	Zimbabwe	ZWE	NaN	NaN	NaN	NaN

774 rows × 60 columns

There is a column in the UN Data for `2015 [YR2015]`, but the instructions say that there isn't data for 2015. I am going to check whether there are values for any countries in the 2015 year column:

In [ ]:

```
pd.set_option('display.max_rows', 150)
df_undata[~df_undata['2015 [YR2015]'].isna()][['Series Name','Country Name','2015 [
```

Out[ ]:

	Series Name	Country Name	2015 [YR2015]
258	Prevalence of undernourishment (% of population)	Afghanistan	26.800000
260	Prevalence of undernourishment (% of population)	Algeria	5.000000
263	Prevalence of undernourishment (% of population)	Angola	14.200000
265	Prevalence of undernourishment (% of population)	Arab World	8.990336
266	Prevalence of undernourishment (% of population)	Argentina	5.000000
267	Prevalence of undernourishment (% of population)	Armenia	5.800000
271	Prevalence of undernourishment (% of population)	Azerbaijan	5.000000
274	Prevalence of undernourishment (% of population)	Bangladesh	16.400000
275	Prevalence of undernourishment (% of population)	Barbados	5.000000
278	Prevalence of undernourishment (% of population)	Belize	6.200000
279	Prevalence of undernourishment (% of population)	Benin	7.500000
282	Prevalence of undernourishment (% of population)	Bolivia	15.900000
284	Prevalence of undernourishment (% of population)	Botswana	24.100000
285	Prevalence of undernourishment (% of population)	Brazil	5.000000
287	Prevalence of undernourishment (% of population)	Brunei Darussalam	5.000000
289	Prevalence of undernourishment (% of population)	Burkina Faso	20.700000
291	Prevalence of undernourishment (% of population)	Cabo Verde	9.400000
292	Prevalence of undernourishment (% of population)	Cambodia	14.200000
293	Prevalence of undernourishment (% of population)	Cameroon	9.900000
295	Prevalence of undernourishment (% of population)	Caribbean small states	7.960086

	Series Name	Country Name	2015 [YR2015]
297	Prevalence of undernourishment (% of population)	Central African Republic	47.700000
299	Prevalence of undernourishment (% of population)	Chad	34.400000
301	Prevalence of undernourishment (% of population)	Chile	5.000000
302	Prevalence of undernourishment (% of population)	China	9.300000
303	Prevalence of undernourishment (% of population)	Colombia	8.800000
306	Prevalence of undernourishment (% of population)	Congo, Rep.	30.500000
307	Prevalence of undernourishment (% of population)	Costa Rica	5.000000
308	Prevalence of undernourishment (% of population)	Cote d'Ivoire	13.300000
310	Prevalence of undernourishment (% of population)	Cuba	5.000000
315	Prevalence of undernourishment (% of population)	Djibouti	15.900000
317	Prevalence of undernourishment (% of population)	Dominican Republic	12.300000
318	Prevalence of undernourishment (% of population)	Early-demographic dividend	13.610680
319	Prevalence of undernourishment (% of population)	East Asia & Pacific	9.746436
320	Prevalence of undernourishment (% of population)	East Asia & Pacific (excluding high income)	9.865943
321	Prevalence of undernourishment (% of population)	East Asia & Pacific (IDA & IBRD countries)	9.467202
322	Prevalence of undernourishment (% of population)	Ecuador	10.900000
323	Prevalence of undernourishment (% of population)	Egypt, Arab Rep.	5.000000
324	Prevalence of undernourishment (% of population)	El Salvador	12.400000
328	Prevalence of undernourishment (% of population)	Ethiopia	32.000000
335	Prevalence of undernourishment (% of population)	Fiji	5.000000

	Series Name	Country Name	2015 [YR2015]
340	Prevalence of undernourishment (% of population)	Gabon	5.000000
341	Prevalence of undernourishment (% of population)	Gambia, The	5.300000
342	Prevalence of undernourishment (% of population)	Georgia	7.400000
344	Prevalence of undernourishment (% of population)	Ghana	5.000000
350	Prevalence of undernourishment (% of population)	Guatemala	15.600000
351	Prevalence of undernourishment (% of population)	Guinea	16.400000
352	Prevalence of undernourishment (% of population)	Guinea-Bissau	20.700000
353	Prevalence of undernourishment (% of population)	Guyana	10.600000
354	Prevalence of undernourishment (% of population)	Haiti	53.400000
355	Prevalence of undernourishment (% of population)	Heavily indebted poor countries (HIPC)	23.943803
357	Prevalence of undernourishment (% of population)	Honduras	12.200000
361	Prevalence of undernourishment (% of population)	India	15.200000
362	Prevalence of undernourishment (% of population)	Indonesia	7.600000
363	Prevalence of undernourishment (% of population)	Iran, Islamic Rep.	5.000000
364	Prevalence of undernourishment (% of population)	Iraq	22.800000
369	Prevalence of undernourishment (% of population)	Jamaica	8.100000
371	Prevalence of undernourishment (% of population)	Jordan	5.000000
372	Prevalence of undernourishment (% of population)	Kazakhstan	5.000000
373	Prevalence of undernourishment (% of population)	Kenya	21.200000
374	Prevalence of undernourishment (% of population)	Kiribati	5.000000

	Series Name	Country Name	2015 [YR2015]
375	Prevalence of undernourishment (% of population)	Korea, Dem. People’s Rep.	41.600000
376	Prevalence of undernourishment (% of population)	Korea, Rep.	5.000000
378	Prevalence of undernourishment (% of population)	Kuwait	5.000000
379	Prevalence of undernourishment (% of population)	Kyrgyz Republic	6.000000
380	Prevalence of undernourishment (% of population)	Lao PDR	18.500000
381	Prevalence of undernourishment (% of population)	Late-demographic dividend	9.056680
382	Prevalence of undernourishment (% of population)	Latin America & Caribbean	7.369545
383	Prevalence of undernourishment (% of population)	Latin America & Caribbean (excluding high income)	7.454325
384	Prevalence of undernourishment (% of population)	Latin America & the Caribbean (IDA & IBRD coun...	7.414424
386	Prevalence of undernourishment (% of population)	Least developed countries: UN classification	22.276434
387	Prevalence of undernourishment (% of population)	Lebanon	5.000000
388	Prevalence of undernourishment (% of population)	Lesotho	11.200000
389	Prevalence of undernourishment (% of population)	Liberia	31.900000
393	Prevalence of undernourishment (% of population)	Low & middle income	12.661516
394	Prevalence of undernourishment (% of population)	Low income	26.088933
395	Prevalence of undernourishment (% of population)	Lower middle income	13.990416
399	Prevalence of undernourishment (% of population)	Madagascar	33.000000
400	Prevalence of undernourishment (% of population)	Malawi	20.700000
401	Prevalence of undernourishment (% of population)	Malaysia	5.000000
402	Prevalence of undernourishment (% of population)	Maldives	5.200000

	Series Name	Country Name	2015 [YR2015]
403	Prevalence of undernourishment (% of population)	Mali	5.000000
406	Prevalence of undernourishment (% of population)	Mauritania	5.600000
407	Prevalence of undernourishment (% of population)	Mauritius	5.000000
408	Prevalence of undernourishment (% of population)	Mexico	5.000000
410	Prevalence of undernourishment (% of population)	Middle East & North Africa	8.200974
411	Prevalence of undernourishment (% of population)	Middle East & North Africa (excluding high inc...	8.672250
412	Prevalence of undernourishment (% of population)	Middle East & North Africa (IDA & IBRD countries)	8.672250
413	Prevalence of undernourishment (% of population)	Middle income	11.315446
416	Prevalence of undernourishment (% of population)	Mongolia	20.500000
418	Prevalence of undernourishment (% of population)	Morocco	5.000000
419	Prevalence of undernourishment (% of population)	Mozambique	25.300000
420	Prevalence of undernourishment (% of population)	Myanmar	14.200000
421	Prevalence of undernourishment (% of population)	Namibia	42.300000
423	Prevalence of undernourishment (% of population)	Nepal	7.800000
427	Prevalence of undernourishment (% of population)	Nicaragua	16.600000
428	Prevalence of undernourishment (% of population)	Niger	9.500000
429	Prevalence of undernourishment (% of population)	Nigeria	7.000000
434	Prevalence of undernourishment (% of population)	Oman	5.000000
436	Prevalence of undernourishment (% of population)	Pacific island small states	6.978035
437	Prevalence of undernourishment (% of population)	Pakistan	22.000000

	Series Name	Country Name	2015 [YR2015]
439	Prevalence of undernourishment (% of population)	Panama	9.500000
441	Prevalence of undernourishment (% of population)	Paraguay	10.400000
442	Prevalence of undernourishment (% of population)	Peru	7.500000
443	Prevalence of undernourishment (% of population)	Philippines	13.500000
447	Prevalence of undernourishment (% of population)	Pre-demographic dividend	18.244150
452	Prevalence of undernourishment (% of population)	Rwanda	31.600000
453	Prevalence of undernourishment (% of population)	Samoa	5.000000
455	Prevalence of undernourishment (% of population)	Sao Tome and Principe	6.600000
456	Prevalence of undernourishment (% of population)	Saudi Arabia	5.000000
457	Prevalence of undernourishment (% of population)	Senegal	10.000000
460	Prevalence of undernourishment (% of population)	Sierra Leone	22.300000
465	Prevalence of undernourishment (% of population)	Small states	14.972683
466	Prevalence of undernourishment (% of population)	Solomon Islands	11.300000
468	Prevalence of undernourishment (% of population)	South Africa	5.000000
469	Prevalence of undernourishment (% of population)	South Asia	16.222532
470	Prevalence of undernourishment (% of population)	South Asia (IDA & IBRD)	16.222532
473	Prevalence of undernourishment (% of population)	Sri Lanka	22.000000
477	Prevalence of undernourishment (% of population)	St. Vincent and the Grenadines	6.200000
478	Prevalence of undernourishment (% of population)	Sub-Saharan Africa	18.522967
479	Prevalence of undernourishment (% of population)	Sub-Saharan Africa (excluding high income)	18.522967

	Series Name	Country Name	2015 [YR2015]
480	Prevalence of undernourishment (% of population)	Sub-Saharan Africa (IDA & IBRD countries)	18.522967
482	Prevalence of undernourishment (% of population)	Suriname	8.000000
483	Prevalence of undernourishment (% of population)	Swaziland	26.800000
487	Prevalence of undernourishment (% of population)	Tajikistan	33.200000
488	Prevalence of undernourishment (% of population)	Tanzania	32.100000
489	Prevalence of undernourishment (% of population)	Thailand	7.400000
490	Prevalence of undernourishment (% of population)	Timor-Leste	26.900000
491	Prevalence of undernourishment (% of population)	Togo	11.400000
493	Prevalence of undernourishment (% of population)	Trinidad and Tobago	7.400000
494	Prevalence of undernourishment (% of population)	Tunisia	5.000000
495	Prevalence of undernourishment (% of population)	Turkey	5.000000
496	Prevalence of undernourishment (% of population)	Turkmenistan	5.000000
499	Prevalence of undernourishment (% of population)	Uganda	25.500000
501	Prevalence of undernourishment (% of population)	United Arab Emirates	5.000000
504	Prevalence of undernourishment (% of population)	Upper middle income	8.174562
505	Prevalence of undernourishment (% of population)	Uruguay	5.000000
506	Prevalence of undernourishment (% of population)	Uzbekistan	5.000000
507	Prevalence of undernourishment (% of population)	Vanuatu	6.400000
508	Prevalence of undernourishment (% of population)	Venezuela, RB	5.000000
509	Prevalence of undernourishment (% of population)	Vietnam	11.000000

	Series Name	Country Name	2015 [YR2015]
512	Prevalence of undernourishment (% of population)	World	10.800000
513	Prevalence of undernourishment (% of population)	Yemen, Rep.	26.100000
514	Prevalence of undernourishment (% of population)	Zambia	47.800000
515	Prevalence of undernourishment (% of population)	Zimbabwe	33.400000

There is data for 2015 for 144 countries in the UN data for undernourishment, but not for the other two features we will be investigating. I assume I should remove this column from this dataframe as well as all the 2015 rows from the VDem dataframe.

In [ ]:

```
print(df_undata.shape)
df_undata_a = df_undata.drop(['2015 [YR2015]'], axis=1)
print(df_undata_a.shape)
df_undata_a.head(2)
```

```
(774, 60)
(774, 59)
```

Out[ ]:

	Series Name	Series Code	Country Name	Country Code	1960 [YR1960]	1961 [YR1961]	1962 [YR1962]	1963 [YR1963]	[YR
0	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	Afghanistan	AFG	0.034844	NaN	NaN	NaN	
1	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	Albania	ALB	0.276291	NaN	NaN	NaN	

Next, I will load the **VDem data**

In [ ]:

```
df_vdem = pd.read_csv(VDem_url)
print(df_vdem.shape)
df_vdem
```

```
(8534, 100)
```

Out[ ]:

	X1	country_name	country_id	country_text_id	year	historical_date	codingstart	gapstar
0	1	Mexico	3	MEX	1960	1960-01-01	1900	NaN
1	2	Mexico	3	MEX	1961	1961-01-01	1900	NaN
2	3	Mexico	3	MEX	1962	1962-01-01	1900	NaN
3	4	Mexico	3	MEX	1963	1963-01-01	1900	NaN
4	5	Mexico	3	MEX	1964	1964-01-01	1900	NaN
...	...	...	...	...	...	...	...	...
8529	8530	Hungary	210	HUN	2008	2008-01-01	1918	NaN
8530	8531	Hungary	210	HUN	2009	2009-01-01	1918	NaN
8531	8532	Hungary	210	HUN	2010	2010-01-01	1918	NaN
8532	8533	Hungary	210	HUN	2011	2011-01-01	1918	NaN
8533	8534	Hungary	210	HUN	2012	2012-01-01	1918	NaN

8534 rows × 100 columns

In [ ]:

```
df_vdem.year.unique()
```

Out[ ]:

```
array([1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970,
       1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981,
       1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992,
       1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
       2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014,
       2015], dtype=int64)
```

Drop all the rows with the year == 2015

In [ ]:

```
print(df_vdem.shape)
# count of rows with year 2015
print("Count of rows with year == 2015: ", df_vdem[df_vdem.year == 2015].shape[0])
df_vdem_a = df_vdem.query('year != 2015')
print(df_vdem_a.shape)
df_vdem_a.head(2)
```

```
(8534, 100)
Count of rows with year == 2015:  76
(8458, 100)
```

Out[ ]:

	X1	country_name	country_id	country_text_id	year	historical_date	codingstart	gapstart	ga
0	1	Mexico	3	MEX	1960	1960-01-01	1900	NaN	
1	2	Mexico	3	MEX	1961	1961-01-01	1900	NaN	

In [ ]:

```
df_vdem_a.year.unique()
```

Out[ ]:
```
array([1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970,
       1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981,
       1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992,
       1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
       2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014],
      dtype=int64)
```

In [ ]:

Part b

The UN data contain certain rows that refer to groups of countries instead of to individual countries. Here’s a list of these non-countries:

In [ ]:
```
noncountries = ['Arab World',  'Caribbean small states',  'Central Europe and the B
    'Early-demographic dividend',  'East Asia & Pacific', 'East Asia & Pacific (exc
    'East Asia & Pacific (IDA & IBRD countries)', 'Euro area', 'Europe & Central As
    'Europe & Central Asia (excluding high income)', 'Europe & Central Asia (IDA &
    'Fragile and conflict affected situations', 'Heavily indebted poor countries (H
    'High income', 'Late-demographic dividend', 'Latin America & Caribbean',
    'Latin America & Caribbean (excluding high income)',
    'Latin America & the Caribbean (IDA & IBRD countries)', 'Least developed countr
    'Low & middle income', 'Low income', 'Lower middle income',
    'Middle East & North Africa', 'Middle East & North Africa (excluding high incom
    'Middle East & North Africa (IDA & IBRD countries)',
    'Middle income', 'North America', 'OECD members',
    'Other small states', 'Pacific island small states', 'Post-demographic dividend
    'Pre-demographic dividend', 'Small states', 'South Asia',
    'South Asia (IDA & IBRD)', 'Sub-Saharan Africa', 'Sub-Saharan Africa (excluding
    'Sub-Saharan Africa (IDA & IBRD countries)', 'Upper middle income', 'World']
```

We can use `.query()` to remove the non-countries from the data, but in this case there are complications due to the space in the name of the column `Country Name` and the use of an external list. So here let's use an alternative method:

First, apply the `.isin(noncountries)` method to the `Country Name` column of the UN data to create a series of values that are `True` if the `Country Name` on a row is one of the non-countries, and `False` otherwise. Second, use the `~` operator to negate the logical values: turn `True` to `False` and vice versa. Finally, pass this logical series to the `.loc[]` attribute of the dataframe to drop the rows that refer to these noncountries from the UN data. (1 point)

(If you wanted to use `.query()`, you would first need to rename `Country Name` to remove the space, then you can use an `@` in front of `noncountries` to refer to the external list. But for this problem follow the instructions listed above.)

Answer Part b

Create the mapping for the filter and apply to `df_undata_a`

In [ ]:

```
# create the inverse of the is in county name list
country_mapping_inv = ~df_undata_a['Country Name'].isin(noncountries)
country_mapping_inv
```

Out[ ]:

```
0       True
1       True
2       True
3       True
4       True
       ...
769     True
770    False
771     True
772     True
773     True
Name: Country Name, Length: 774, dtype: bool
```

In [ ]:

```
print(df_undata_a.shape)
print("Number of Trues: ", country_mapping_inv.sum())
df_undata_b = df_undata_a.loc[country_mapping_inv]
print(df_undata_b.shape)
```

```
(774, 59)
Number of Trues:  651
(651, 59)
```

Part c

Reshape the UN data to move the years from the columns to the rows. (Once the years are in the rows, they will have values such as "1960 [YR1960]".) (2 points)

Answer Part c

Pivot the `df_undata_b`

In [ ]:

```
df_undata_b.head(2)
```

Out[ ]:

	Series Name	Series Code	Country Name	Country Code	1960 [YR1960]	1961 [YR1961]	1962 [YR1962]	1963 [YR1963]	[YR
0	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	Afghanistan	AFG	0.034844	NaN	NaN	NaN	
1	Physicians (per 1,000 people)	SH.MED.PHYS.ZS	Albania	ALB	0.276291	NaN	NaN	NaN	

In [ ]:

```
#df_undata_b['Series Name'].unique()
series_names
```

Out[ ]:

```
array(['Physicians (per 1,000 people)',
       'Prevalence of undernourishment (% of population)',
       'Health expenditure per capita (current US$)'], dtype=object)
```

In [ ]:

```
# get the unique series names
series_names = df_undata_b['Series Name'].unique()
country_names = df_undata_b['Country Name'].unique()
year_cols = [col for col in df_undata_b.columns if col.endswith(']')]

print(country_names.shape)
print(len(year_cols))
print(country_names.shape[0] * len(year_cols))
```

```
(217,)
55
11935
```

In [ ]:

```
df_undata_c_final = pd.DataFrame(country_names, columns=['Country Name'])
df_undata_c_final.set_index('Country Name', inplace=True)
# for each series name I am going to use pandas.melt to change the columns to rows
for series in series_names:
    print(series)

    # change columns to rows
    year_cols = [col for col in df_undata_b.columns if col.endswith(']')]

    df_series = df_undata_b[df_undata_b['Series Name'] == series]
    #print(df_series.head(1))

    # using pandas.melt (https://pandas.pydata.org/docs/reference/api/pandas.melt.h
    df_series_c = df_series.melt(id_vars=['Country Name'], value_vars=year_cols, va
    #print(df_series_c.head(1))


    # if the first series, then keep the index only Country Name
    if series == series_names[0]:
        # set the index on df_series_c to Country Name
        df_series_c.set_index('Country Name', inplace=True)
        # join df_series_c to df_undata_c_final by Country Name
        df_undata_c_final = df_undata_c_final.join(df_series_c, on=['Country Name']
    else:
        df_undata_c_final.reset_index(inplace=True)
        df_undata_c_final.set_index(['Country Name','year'], inplace=True)
        # set the index on df_series_c to Country Name
        df_series_c.set_index(['Country Name','year'], inplace=True)
        # join df_series_c to df_undata_c_final by Country Name
        df_undata_c_final = df_undata_c_final.join(df_series_c, on=['Country Name',

# drop the index from df_undata_c_final
df_undata_c_final.reset_index(inplace=True)
df_undata_c_final
```

```
Physicians (per 1,000 people)
Prevalence of undernourishment (% of population)
Health expenditure per capita (current US$)
```

Out[ ]:

	Country Name	year	Physicians (per 1,000 people)	Prevalence of undernourishment (% of population)	Health expenditure per capita (current US$)
0	Afghanistan	1960 [YR1960]	0.034844	NaN	NaN
1	Afghanistan	1961 [YR1961]	NaN	NaN	NaN
2	Afghanistan	1962 [YR1962]	NaN	NaN	NaN
3	Afghanistan	1963 [YR1963]	NaN	NaN	NaN
4	Afghanistan	1964 [YR1964]	NaN	NaN	NaN
...	...	...	...	...	...
11930	Zimbabwe	2010 [YR2010]	0.068000	34.7	36.362794
11931	Zimbabwe	2011 [YR2011]	0.083000	33.5	48.469580
11932	Zimbabwe	2012 [YR2012]	NaN	33.2	57.253763
11933	Zimbabwe	2013 [YR2013]	NaN	33.5	62.309228
11934	Zimbabwe	2014 [YR2014]	NaN	34.0	57.710452

11935 rows × 5 columns

In [ ]:

```
# just looking at some random rows to get an idea of the data
df_undata_c_final.iloc[1030:1040]
```

Out[ ]:

	Country Name	year	Physicians (per 1,000 people)	Prevalence of undernourishment (% of population)	Health expenditure per capita (current US$)
1030	Belgium	2000 [YR2000]	3.900	NaN	1845.338117
1031	Belgium	2001 [YR2001]	3.900	NaN	1878.367001
1032	Belgium	2002 [YR2002]	3.900	NaN	2075.073771
1033	Belgium	2003 [YR2003]	NaN	NaN	2800.312474
1034	Belgium	2004 [YR2004]	NaN	NaN	3242.842301
1035	Belgium	2005 [YR2005]	2.073	NaN	3339.718116
1036	Belgium	2006 [YR2006]	2.132	NaN	3489.806735
1037	Belgium	2007 [YR2007]	NaN	NaN	4017.788797
1038	Belgium	2008 [YR2008]	2.987	NaN	4565.948425
1039	Belgium	2009 [YR2009]	NaN	NaN	4574.694810

In [ ]:

```
# find rows where there are no NaNs
df_undata_c_final[~df_undata_c_final.isna().any(axis=1)]
```

Out[ ]:

	Country Name	year	Physicians (per 1,000 people)	Prevalence of undernourishment (% of population)	Health expenditure per capita (current US$)
45	Afghanistan	2005 [YR2005]	0.200	35.2	21.895803
46	Afghanistan	2006 [YR2006]	0.136	32.3	22.181409
47	Afghanistan	2007 [YR2007]	0.146	29.9	26.988947
48	Afghanistan	2008 [YR2008]	0.145	27.7	33.805189
49	Afghanistan	2009 [YR2009]	0.175	26.0	43.525440
...	...	...	...	...	...
11927	Zimbabwe	2007 [YR2007]	0.051	38.6	17.804017
11928	Zimbabwe	2008 [YR2008]	0.056	37.3	16.213711
11929	Zimbabwe	2009 [YR2009]	0.062	36.0	37.207024
11930	Zimbabwe	2010 [YR2010]	0.068	34.7	36.362794
11931	Zimbabwe	2011 [YR2011]	0.083	33.5	48.469580

972 rows × 5 columns

Part d

Rename the `variable` column to `year`. Then use string methods to remove the ends such as "[YR1960]" from the values of the new `year` column and convert the column to an integer data type.

Also, for whatever reason, real world data often contains multiple variables that are just different representations of the same information. In this case, the `Series Name` and `Series Code` variables tell us exactly the same thing, and the `Country Name` and `Country Code` variables tell us exactly the same thing. Unless I have a very good reason to keep both, I generally prefer to drop variables that are redundant and coded in a less helpful way. So drop `Series Code` and `Country Code`. (2 points)

Answer Part d

- I already have renamed the column to `year` in part c.
- I already took care of the country code and series code in part c.

In [ ]:

```
# make the year column the first 4 characters of the year column
df_undata_c_final.year = df_undata_c_final.year.str[0:4]
# then turn it into an int column
df_undata_c_final.year = df_undata_c_final.year.astype(int)
```

In [ ]:

```
df_undata_c_final.dtypes
```

Out[ ]:

```
country_name          object
year                   int32
phys_per_1000        float64
undernour_percpop    float64
health_dol_percap    float64
dtype: object
```

Part e

Reshape the data to move the values of `Series Name` to separate columns. Make sure all of the columns exist in the dataframe after reshaping and are not stored in a row index or multi-index. Then rename the columns so that all of the columns have concise and descriptive names. (2 points)

Answer part 3: renaming columns

In [ ]:

```
df_undata_c_final.columns = ['country_name','year','phys_per_1000','undernour_percp
df_undata_c_final
```

Out[ ]:

	country_name	year	phys_per_1000	undernour_percpop	health_dol_percap
0	Afghanistan	1960	0.034844	NaN	NaN
1	Afghanistan	1961	NaN	NaN	NaN
2	Afghanistan	1962	NaN	NaN	NaN
3	Afghanistan	1963	NaN	NaN	NaN
4	Afghanistan	1964	NaN	NaN	NaN
...	...	...	...	...	...
11930	Zimbabwe	2010	0.068000	34.7	36.362794
11931	Zimbabwe	2011	0.083000	33.5	48.469580
11932	Zimbabwe	2012	NaN	33.2	57.253763
11933	Zimbabwe	2013	NaN	33.5	62.309228
11934	Zimbabwe	2014	NaN	34.0	57.710452

11935 rows × 5 columns

Part f

Next we are going to join the cleaned UN data with the VDem data. In a perfect world, both datasets would include a shared numeric country ID field that we can use to match countries in one dataset to countries in the other. Unfortunately the UN data identifies the countries only by name. Worse still, while there is a big overlap the two datasets cover different sets of countries.

First decide whether this merge is a one-to-one, one-to-many, many-to-one, or many-to-many merge and describe your rationale in words.

Then perform a test merge that checks whether your expectation that the merge is one-to-one, one-to-many, many-to-one, or many-to-many is confirmed, and reports whether each row is matched, appears only in the UN data, or appears only in the VDem data. Use the `.unique()` or `.value_counts()` method to display the names of the countries that are not matched. (2 points)

Answer 1f

I expect this to be a one-to-one merge. My rationale is that the **"id"**/or "superkey" for each table are the "country_name" and "year" columns combined.

First, I wanted to get an idea of the intersect count, and difference counts for countries in each of the two dataframes:

The First 5 Rows of the VDem Table

In [ ]:

```
df_vdem_a.head()
```

Out[ ]:

	X1	country_name	country_id	country_text_id	year	historical_date	codingstart	gapstart	ga
0	1	Mexico	3	MEX	1960	1960-01-01	1900	NaN	
1	2	Mexico	3	MEX	1961	1961-01-01	1900	NaN	
2	3	Mexico	3	MEX	1962	1962-01-01	1900	NaN	
3	4	Mexico	3	MEX	1963	1963-01-01	1900	NaN	
4	5	Mexico	3	MEX	1964	1964-01-01	1900	NaN	

What are dtypes of the VDem table superkey?

In [ ]:

```
df_vdem_a[['country_name','year']].dtypes
```

Out[ ]:

```
country_name    object
year             int64
dtype: object
```

Total Number of Rows In the VDem Table

In [ ]:

```
df_vdem_a.shape
```

Out[ ]:

```
(8458, 100)
```

The Years Per Country Before Merging In the VDem Table

In [ ]:

```
pd.set_option('display.max_rows', 172)
df_vdem_a[['country_name','year']].groupby('country_name').count()
```

Out[ ]:

	year
country_name	
Afghanistan	55
Albania	53
Algeria	55
Angola	53
Argentina	55
Armenia	25
Australia	55
Austria	53
Azerbaijan	25
Bangladesh	44
Barbados	55
Belarus	25
Belgium	55
Benin	55
Bhutan	55
Bolivia	55
Bosnia and Herzegovina	23
Botswana	55
Brazil	55
Bulgaria	55
Burkina Faso	55
Burma_Myanmar	55
Burundi	55
Cambodia	55
Cameroon	54
Canada	55
Cape Verde	55
Central African Republic	53
Chad	53
Chile	55
China	55
Colombia	55

	year
country_name	
Comoros	53
Congo_Democratic Republic of	53
Congo_Republic of the	53
Costa Rica	55
Croatia	21
Cuba	55
Cyprus	53
Czech Republic	53
Denmark	55
Djibouti	53
Dominican Republic	55
East Timor	55
Ecuador	53
Egypt	55
El Salvador	55
Eritrea	55
Estonia	24
Ethiopia	55
Fiji	55
Finland	55
France	53
Gabon	53
Gambia	53
Georgia	25
German Democratic Republic	31
Germany	55
Ghana	55
Greece	53
Guatemala	53
Guinea	53
Guinea-Bissau	53
Guyana	55

	year
country_name	
Haiti	53
Honduras	53
Hungary	53
Iceland	53
India	55
Indonesia	55
Iran	55
Iraq	55
Ireland	53
Israel	53
Italy	53
Ivory Coast	53
Jamaica	53
Japan	55
Jordan	55
Kazakhstan	25
Kenya	55
Korea_North	53
Korea_South	55
Kosovo	16
Kyrgyzstan	25
Laos	53
Latvia	24
Lebanon	55
Lesotho	53
Liberia	53
Libya	55
Lithuania	24
Macedonia	24
Madagascar	53
Malawi	55
Malaysia	53

	year
country_name	
Maldives	55
Mali	53
Mauritania	53
Mauritius	55
Mexico	55
Moldova	25
Mongolia	55
Montenegro	14
Morocco	55
Mozambique	55
Namibia	55
Nepal	55
Netherlands	55
New Zealand	53
Nicaragua	53
Niger	53
Nigeria	55
Norway	55
Pakistan	55
Palestine_Gaza	14
Palestine_West_Bank	47
Panama	53
Papua New Guinea	55
Paraguay	55
Peru	55
Philippines	55
Poland	55
Portugal	55
Qatar	55
Romania	55
Russia	55
Rwanda	55

	year
country_name	
Sao Tome and Principe	53
Saudi Arabia	53
Senegal	53
Serbia	53
Seychelles	53
Sierra Leone	53
Slovakia	19
Slovenia	26
Solomon Islands	55
Somalia	55
Somaliland	23
South Africa	55
South Sudan	4
South Yemen	31
Spain	55
Sri Lanka	55
Sudan	55
Suriname	55
Swaziland	53
Sweden	55
Switzerland	55
Syria	55
Taiwan	55
Tajikistan	25
Tanzania	55
Thailand	55
Togo	53
Trinidad and Tobago	53
Tunisia	55
Turkey	55
Turkmenistan	23
Uganda	55

	year
country_name	
Ukraine	25
United Kingdom	53
United States	55
Uruguay	55
Uzbekistan	25
Vanuatu	55
Venezuela	53
Vietnam_Democratic Republic of	53
Vietnam_Republic of	16
Yemen	55
Zambia	55
Zimbabwe	55

The First 5 Rows of the UN Table

In [ ]:

```
df_undata_c_final.head()
```

Out[ ]:

	country_name	year	phys_per_1000	undernour_percpop	health_dol_percap
0	Afghanistan	1960 [YR1960]	0.034844	NaN	NaN
1	Afghanistan	1961 [YR1961]	NaN	NaN	NaN
2	Afghanistan	1962 [YR1962]	NaN	NaN	NaN
3	Afghanistan	1963 [YR1963]	NaN	NaN	NaN
4	Afghanistan	1964 [YR1964]	NaN	NaN	NaN

What are the dtypes of the UN Table superkey?

In [ ]:

```
df_undata_c_final[['country_name','year']].dtypes
```

Out[ ]:

```
country_name    object
year             int32
dtype: object
```

Total Number of Rows in the UN Table

In [ ]:

```
df_undata_c_final.shape
```

Out[ ]:

```
(11935, 5)
```

The Years Per Country Before Merging In the UN Table

Unlike the VDem table, these are all 55. **We will loose some data with an inner join because of missing temporal data.**

```
In [ ]: pd.set_option('display.max_rows', 217)
        df_undata_c_final[['country_name','year']].groupby('country_name').count()
```

Out[ ]:

	year
country_name	
Afghanistan	55
Albania	55
Algeria	55
American Samoa	55
Andorra	55
Angola	55
Antigua and Barbuda	55
Argentina	55
Armenia	55
Aruba	55
Australia	55
Austria	55
Azerbaijan	55
Bahamas, The	55
Bahrain	55
Bangladesh	55
Barbados	55
Belarus	55
Belgium	55
Belize	55
Benin	55
Bermuda	55
Bhutan	55
Bolivia	55
Bosnia and Herzegovina	55
Botswana	55
Brazil	55
British Virgin Islands	55
Brunei Darussalam	55
Bulgaria	55
Burkina Faso	55
Burundi	55

	year
country_name	
Cabo Verde	55
Cambodia	55
Cameroon	55
Canada	55
Cayman Islands	55
Central African Republic	55
Chad	55
Channel Islands	55
Chile	55
China	55
Colombia	55
Comoros	55
Congo, Dem. Rep.	55
Congo, Rep.	55
Costa Rica	55
Cote d'Ivoire	55
Croatia	55
Cuba	55
Curacao	55
Cyprus	55
Czech Republic	55
Denmark	55
Djibouti	55
Dominica	55
Dominican Republic	55
Ecuador	55
Egypt, Arab Rep.	55
El Salvador	55
Equatorial Guinea	55
Eritrea	55
Estonia	55
Ethiopia	55

	year
country_name	
Faroe Islands	55
Fiji	55
Finland	55
France	55
French Polynesia	55
Gabon	55
Gambia, The	55
Georgia	55
Germany	55
Ghana	55
Gibraltar	55
Greece	55
Greenland	55
Grenada	55
Guam	55
Guatemala	55
Guinea	55
Guinea-Bissau	55
Guyana	55
Haiti	55
Honduras	55
Hong Kong SAR, China	55
Hungary	55
Iceland	55
India	55
Indonesia	55
Iran, Islamic Rep.	55
Iraq	55
Ireland	55
Isle of Man	55
Israel	55
Italy	55

	year
country_name	
Jamaica	55
Japan	55
Jordan	55
Kazakhstan	55
Kenya	55
Kiribati	55
Korea, Dem. People’s Rep.	55
Korea, Rep.	55
Kosovo	55
Kuwait	55
Kyrgyz Republic	55
Lao PDR	55
Latvia	55
Lebanon	55
Lesotho	55
Liberia	55
Libya	55
Liechtenstein	55
Lithuania	55
Luxembourg	55
Macao SAR, China	55
Macedonia, FYR	55
Madagascar	55
Malawi	55
Malaysia	55
Maldives	55
Mali	55
Malta	55
Marshall Islands	55
Mauritania	55
Mauritius	55
Mexico	55

	year
country_name	
Micronesia, Fed. Sts.	55
Moldova	55
Monaco	55
Mongolia	55
Montenegro	55
Morocco	55
Mozambique	55
Myanmar	55
Namibia	55
Nauru	55
Nepal	55
Netherlands	55
New Caledonia	55
New Zealand	55
Nicaragua	55
Niger	55
Nigeria	55
Northern Mariana Islands	55
Norway	55
Oman	55
Pakistan	55
Palau	55
Panama	55
Papua New Guinea	55
Paraguay	55
Peru	55
Philippines	55
Poland	55
Portugal	55
Puerto Rico	55
Qatar	55
Romania	55

	year
country_name	
Russian Federation	55
Rwanda	55
Samoa	55
San Marino	55
Sao Tome and Principe	55
Saudi Arabia	55
Senegal	55
Serbia	55
Seychelles	55
Sierra Leone	55
Singapore	55
Sint Maarten (Dutch part)	55
Slovak Republic	55
Slovenia	55
Solomon Islands	55
Somalia	55
South Africa	55
South Sudan	55
Spain	55
Sri Lanka	55
St. Kitts and Nevis	55
St. Lucia	55
St. Martin (French part)	55
St. Vincent and the Grenadines	55
Sudan	55
Suriname	55
Swaziland	55
Sweden	55
Switzerland	55
Syrian Arab Republic	55
Tajikistan	55
Tanzania	55

	year
country_name	
Thailand	55
Timor-Leste	55
Togo	55
Tonga	55
Trinidad and Tobago	55
Tunisia	55
Turkey	55
Turkmenistan	55
Turks and Caicos Islands	55
Tuvalu	55
Uganda	55
Ukraine	55
United Arab Emirates	55
United Kingdom	55
United States	55
Uruguay	55
Uzbekistan	55
Vanuatu	55
Venezuela, RB	55
Vietnam	55
Virgin Islands (U.S.)	55
West Bank and Gaza	55
Yemen, Rep.	55
Zambia	55
Zimbabwe	55

In [ ]:

```
vdem_country_names = df_vdem_a.country_name.unique()
un_country_names = df_undata_c_final.country_name.sort_values().unique()
print('shape vdem country names: ', vdem_country_names.shape)
print('shape un country names: ', un_country_names.shape)

# which countries are in vdem but not in un
diff_vdem_un = np.setdiff1d(vdem_country_names, un_country_names)
print('len diff_vdem_un: ', len(diff_vdem_un))

# which countries are in un but not in vdem
```

```
diff_un_vdem = np.setdiff1d(un_country_names, vdem_country_names)
print('len diff_un_vdem: ', len(diff_un_vdem))

# which countries are in both
intersect_vdem_un = np.intersect1d(vdem_country_names, un_country_names)
print('len intersect_vdem_un: ', len(intersect_vdem_un))
```

```
shape vdem country names:  (172,)
shape un country names:  (217,)
len diff_vdem_un:  27
len diff_un_vdem:  72
len intersect_vdem_un:  145
```

It looks like there are:

- **27** countries in the VDem dataset that are not in the UN dataset
- **72** countries in the UN dataset that are not in the VDem dataset

What are these countries?

In [ ]:

```
# in VDem, but not in UN
diff_vdem_un
```

Out[ ]:

```
array(['Burma_Myanmar', 'Cape Verde', 'Congo_Democratic Republic of',
       'Congo_Republic of the', 'East Timor', 'Egypt', 'Gambia',
       'German Democratic Republic', 'Iran', 'Ivory Coast', 'Korea_North',
       'Korea_South', 'Kyrgyzstan', 'Laos', 'Macedonia', 'Palestine_Gaza',
       'Palestine_West_Bank', 'Russia', 'Slovakia', 'Somaliland',
       'South Yemen', 'Syria', 'Taiwan', 'Venezuela',
       'Vietnam_Democratic Republic of', 'Vietnam_Republic of', 'Yemen'],
      dtype=object)
```

In [ ]:

```
# in UN, but not in VDem
diff_un_vdem
```

Out[ ]:

```
array(['American Samoa', 'Andorra', 'Antigua and Barbuda', 'Aruba',
       'Bahamas, The', 'Bahrain', 'Belize', 'Bermuda',
       'British Virgin Islands', 'Brunei Darussalam', 'Cabo Verde',
       'Cayman Islands', 'Channel Islands', 'Congo, Dem. Rep.',
       'Congo, Rep.', "Cote d'Ivoire", 'Curacao', 'Dominica',
       'Egypt, Arab Rep.', 'Equatorial Guinea', 'Faroe Islands',
       'French Polynesia', 'Gambia, The', 'Gibraltar', 'Greenland',
       'Grenada', 'Guam', 'Hong Kong SAR, China', 'Iran, Islamic Rep.',
       'Isle of Man', 'Kiribati', 'Korea, Dem. People’s Rep.',
       'Korea, Rep.', 'Kuwait', 'Kyrgyz Republic', 'Lao PDR',
       'Liechtenstein', 'Luxembourg', 'Macao SAR, China',
       'Macedonia, FYR', 'Malta', 'Marshall Islands',
       'Micronesia, Fed. Sts.', 'Monaco', 'Myanmar', 'Nauru',
       'New Caledonia', 'Northern Mariana Islands', 'Oman', 'Palau',
       'Puerto Rico', 'Russian Federation', 'Samoa', 'San Marino',
       'Singapore', 'Sint Maarten (Dutch part)', 'Slovak Republic',
       'St. Kitts and Nevis', 'St. Lucia', 'St. Martin (French part)',
       'St. Vincent and the Grenadines', 'Syrian Arab Republic',
       'Timor-Leste', 'Tonga', 'Turks and Caicos Islands', 'Tuvalu',
       'United Arab Emirates', 'Venezuela, RB', 'Vietnam',
       'Virgin Islands (U.S.)', 'West Bank and Gaza', 'Yemen, Rep.'],
      dtype=object)
```

I started re-coding these, and then saw that this was part of the **Part g** exercise, so I will wait until then.

Now, I will do the requested test merge:

- perform a test merge
  - checks one-to-one is confirmed
  - reports whether each row is:
    - matched
    - appears only in the UN data
    - or appears only in the VDem data
- Use the `.unique()` or `.value_counts()` method to display the names of the countries that are not matched.
- (2 points)

In [ ]:

```python
test_merge = pd.merge(df_undata_c_final, df_vdem_a, on=['country_name','year'], how
test_merge.head()
```

Out[ ]:

	country_name	year	phys_per_1000	undernour_percpop	health_dol_percap	X1	country_id
0	Afghanistan	1960	0.034844	NaN	NaN	1583.0	36.0
1	Afghanistan	1961	NaN	NaN	NaN	1584.0	36.0
2	Afghanistan	1962	NaN	NaN	NaN	1585.0	36.0
3	Afghanistan	1963	NaN	NaN	NaN	1586.0	36.0
4	Afghanistan	1964	NaN	NaN	NaN	1587.0	36.0

In [ ]:

```
pd.DataFrame(test_merge[['country_name','matched']].value_counts()).query('matched
```

Out[ ]:

		0
country_name	**matched**	
Greenland	**left_only**	55
Malta	**left_only**	55
Marshall Islands	**left_only**	55
Micronesia, Fed. Sts.	**left_only**	55
Monaco	**left_only**	55
Myanmar	**left_only**	55
Nauru	**left_only**	55
New Caledonia	**left_only**	55
Northern Mariana Islands	**left_only**	55
Oman	**left_only**	55
Palau	**left_only**	55
Macedonia, FYR	**left_only**	55
Macao SAR, China	**left_only**	55
Guam	**left_only**	55
Hong Kong SAR, China	**left_only**	55
Iran	**right_only**	55
Iran, Islamic Rep.	**left_only**	55
Isle of Man	**left_only**	55
Luxembourg	**left_only**	55
Kiribati	**left_only**	55
Korea, Dem. People’s Rep.	**left_only**	55
Korea, Rep.	**left_only**	55
Korea_South	**right_only**	55
Kuwait	**left_only**	55
Kyrgyz Republic	**left_only**	55
Lao PDR	**left_only**	55
Liechtenstein	**left_only**	55
Syrian Arab Republic	**left_only**	55
Taiwan	**right_only**	55
Timor-Leste	**left_only**	55
Tonga	**left_only**	55
Turks and Caicos Islands	**left_only**	55

		0
country_name	**matched**	
Tuvalu	**left_only**	55
United Arab Emirates	**left_only**	55
Venezuela, RB	**left_only**	55
Vietnam	**left_only**	55
Virgin Islands (U.S.)	**left_only**	55
West Bank and Gaza	**left_only**	55
Yemen	**right_only**	55
Yemen, Rep.	**left_only**	55
Syria	**right_only**	55
Sint Maarten (Dutch part)	**left_only**	55
Puerto Rico	**left_only**	55
Russia	**right_only**	55
Russian Federation	**left_only**	55
Samoa	**left_only**	55
San Marino	**left_only**	55
Singapore	**left_only**	55
Slovak Republic	**left_only**	55
St. Kitts and Nevis	**left_only**	55
St. Lucia	**left_only**	55
St. Martin (French part)	**left_only**	55
St. Vincent and the Grenadines	**left_only**	55
Grenada	**left_only**	55
Burma_Myanmar	**right_only**	55
Faroe Islands	**left_only**	55
Cabo Verde	**left_only**	55
Equatorial Guinea	**left_only**	55
Cape Verde	**right_only**	55
Cayman Islands	**left_only**	55
Egypt, Arab Rep.	**left_only**	55
Brunei Darussalam	**left_only**	55
Channel Islands	**left_only**	55
Egypt	**right_only**	55

		0
country_name	**matched**	
East Timor	**right_only**	55
Congo, Dem. Rep.	**left_only**	55
Congo, Rep.	**left_only**	55
Dominica	**left_only**	55
Cote d'Ivoire	**left_only**	55
French Polynesia	**left_only**	55
Curacao	**left_only**	55
British Virgin Islands	**left_only**	55
American Samoa	**left_only**	55
Aruba	**left_only**	55
Andorra	**left_only**	55
Bahamas, The	**left_only**	55
Bahrain	**left_only**	55
Gambia, The	**left_only**	55
Antigua and Barbuda	**left_only**	55
Gibraltar	**left_only**	55
Bermuda	**left_only**	55
Belize	**left_only**	55
Vietnam_Democratic Republic of	**right_only**	53
Congo_Republic of the	**right_only**	53
Venezuela	**right_only**	53
Congo_Democratic Republic of	**right_only**	53
Laos	**right_only**	53
Gambia	**right_only**	53
Ivory Coast	**right_only**	53
Korea_North	**right_only**	53
South Sudan	**left_only**	51
Palestine_West_Bank	**right_only**	47
Montenegro	**left_only**	41
Kosovo	**left_only**	39
Croatia	**left_only**	34
Turkmenistan	**left_only**	32

		0
country_name	**matched**	
Bosnia and Herzegovina	**left_only**	32
Latvia	**left_only**	31
Estonia	**left_only**	31
South Yemen	**right_only**	31
German Democratic Republic	**right_only**	31
Lithuania	**left_only**	31
Moldova	**left_only**	30
Tajikistan	**left_only**	30
Kazakhstan	**left_only**	30
Ukraine	**left_only**	30
Armenia	**left_only**	30
Uzbekistan	**left_only**	30
Georgia	**left_only**	30
Azerbaijan	**left_only**	30
Belarus	**left_only**	30
Slovenia	**left_only**	29
Kyrgyzstan	**right_only**	25
Macedonia	**right_only**	24
Somaliland	**right_only**	23
Slovakia	**right_only**	19
Vietnam_Republic of	**right_only**	16
Palestine_Gaza	**right_only**	14
Bangladesh	**left_only**	11
Nicaragua	**left_only**	2
Hungary	**left_only**	2
Comoros	**left_only**	2
Malaysia	**left_only**	2
Greece	**left_only**	2
United Kingdom	**left_only**	2
Mali	**left_only**	2
Niger	**left_only**	2
Mauritania	**left_only**	2

		0
country_name	**matched**	
New Zealand	**left_only**	2
Madagascar	**left_only**	2
Honduras	**left_only**	2
Djibouti	**left_only**	2
Haiti	**left_only**	2
France	**left_only**	2
Guinea-Bissau	**left_only**	2
Angola	**left_only**	2
Cyprus	**left_only**	2
Guinea	**left_only**	2
Guatemala	**left_only**	2
Austria	**left_only**	2
Serbia	**left_only**	2
Jamaica	**left_only**	2
Panama	**left_only**	2
Italy	**left_only**	2
Senegal	**left_only**	2
Israel	**left_only**	2
Gabon	**left_only**	2
Swaziland	**left_only**	2
Saudi Arabia	**left_only**	2
Ireland	**left_only**	2
Albania	**left_only**	2
Sao Tome and Principe	**left_only**	2
Seychelles	**left_only**	2
Lesotho	**left_only**	2
Liberia	**left_only**	2
Central African Republic	**left_only**	2
Chad	**left_only**	2
Togo	**left_only**	2
Sierra Leone	**left_only**	2
Trinidad and Tobago	**left_only**	2

		0
country_name	**matched**	
Iceland	**left_only**	2
Ecuador	**left_only**	2
Czech Republic	**left_only**	2
Cameroon	**left_only**	1

Part g

There are many unmatched rows in this merge. There are three reasons why rows failed to match:

- Differences in geographical coverage: for example, the VDem data includes Taiwan, but the UN data does not
- Differences in time coverage: for example, the UN data includes records for France every year from 1970 through 2014, and VDem includes rows for France from 1960 to 2012, leaving 12 rows for France without matching years
- Differences in spelling: for example, South Korea is called "Korea, Rep." in the UN data and "Korea_South" in the VDem data.

We can't do anything about differences in geographic or temporal coverage. But we can recode some country names to account for differences in spelling and to match more rows that should match. Here is a list of differently spelled countries:

1. "Burma_Myanmar" in VDem is "Myanmar" in the UN data
2. "Cape Verde" in VDem is "Cabo Verde" in the UN data
3. "Congo_Democratic Republic of" in VDem is "Congo, Dem. Rep." in the UN data
4. "Congo_Republic of the" in VDem is "Congo, Rep." in the UN data
5. "East Timor" in VDem is "Timor-Leste" in the UN data
6. "Egypt" in VDem is "Egypt, Arab Rep." in the UN data
7. "Gambia" in VDem is "Gambia, The" in the UN data
8. "Iran" in VDem is "Iran, Islamic Rep." in the UN data
9. "Ivory Coast" in VDem is "Cote d’Ivoire" in the UN data
10. "Korea_North" in VDem is "Korea, Dem. People’s Rep." in the UN data
11. "Korea_South" in VDem is "Korea, Rep." in the UN data
12. "Kyrgyzstan" in VDem is "Kyrgyz Republic" in the UN data
13. "Laos" in VDem is "Lao PDR" in the UN data
14. "Macedonia" in VDem is "Macedonia, FYR" in the UN data
15. "Palestine_West_Bank" in VDem is "West Bank and Gaza" in the UN Data (there is also "Palestine_Gaza" in VDem, but since the UN combines data for the West Bank and Gaza, let's just use "Palestine_West_Bank" for this assignment)
16. "Russia" in VDem is "Russian Federation" in the UN data

17. "Slovakia" in VDem is "Slovak Republic" in the UN data
18. "Syria" in VDem is "Syrian Arab Republic" in the UN data
19. "Venezuela" in VDem is "Venezuela, RB" in the UN data
20. "Vietnam_Democratic Republic of" in VDem is "Vietnam" in the UN data
21. "Yemen" in VDem is "Yemen, Rep." in the UN data

Recode the country names listed above in one of the two dataframes to match the names in the other dataframe. Then perform an inner join of the two dataframes. Some rows will be dropped because of differences in coverage, but no rows will be dropped because of differences in spelling. (2 points)

In [ ]:

```
# recoding the country names to match
df_vdem_g = df_vdem_a.copy()
# in the df_vdem_g dataframe where teh country name is 'Burma_Myanmar' replace with
df_vdem_g.country_name = df_vdem_g.country_name.str.replace('Burma_Myanmar', 'Myanm
df_vdem_g.country_name = df_vdem_g.country_name.str.replace('Congo_Republic of the'
df_vdem_g.country_name = df_vdem_g.country_name.str.replace('Gambia', 'Gambia, The'
df_vdem_g.country_name = df_vdem_g.country_name.str.replace('Korea_North', 'Korea,
df_vdem_g.country_name = df_vdem_g.country_name.str.replace('Laos', 'Lao PDR').repl
df_vdem_g.country_name = df_vdem_g.country_name.str.replace('Russia', 'Russian Fede
df_vdem_g.country_name = df_vdem_g.country_name.str.replace('Venezuela', 'Venezuela
```

In [ ]:

```
df_vdem_g.query('country_name == "Myanmar"').shape
```

Out[ ]:

```
(55, 100)
```

In [ ]:

```
final_merge = pd.merge(df_undata_c_final, df_vdem_a, on=['country_name','year'], ho
print(final_merge.shape)
final_merge.head(10)
```

```
(7254, 103)
```

Out[ ]:

	country_name	year	phys_per_1000	undernour_percpop	health_dol_percap	X1	country_id
0	Afghanistan	1960	0.034844	NaN	NaN	1583	36
1	Afghanistan	1961	NaN	NaN	NaN	1584	36
2	Afghanistan	1962	NaN	NaN	NaN	1585	36
3	Afghanistan	1963	NaN	NaN	NaN	1586	36
4	Afghanistan	1964	NaN	NaN	NaN	1587	36
5	Afghanistan	1965	0.063428	NaN	NaN	1588	36
6	Afghanistan	1966	NaN	NaN	NaN	1589	36
7	Afghanistan	1967	NaN	NaN	NaN	1590	36
8	Afghanistan	1968	NaN	NaN	NaN	1591	36
9	Afghanistan	1969	NaN	NaN	NaN	1592	36

In [ ]:

```
final_merge.tail()
```

Out[ ]:

	country_name	year	phys_per_1000	undernour_percpop	health_dol_percap	X1	country_i
7249	Zimbabwe	2010	0.068	34.7	36.362794	3035	6
7250	Zimbabwe	2011	0.083	33.5	48.469580	3036	6
7251	Zimbabwe	2012	NaN	33.2	57.253763	3037	6
7252	Zimbabwe	2013	NaN	33.5	62.309228	3038	6
7253	Zimbabwe	2014	NaN	34.0	57.710452	3039	6

Problem 2

Kickstarter is a website in which people can pledge financial support for creative projects. Patrons are only charged if a project raises enough money to meet a pre-specified goal, and projects can offer items as "rewards" for patrons who contribute at particular levels. One interesting aspect of Kickstarter is the ability to search projects by "ending soon". If you have a few dollars to spare and want to feel like a hero, you can swoop in at the last minute to contribute enough for a project to meet its goal.

Cathie So created a project on Kaggle in which she scraped Kickstarter and collected data on 4000 live projects (projects that were currently collecting pledges from patrons) as of October 10, 2016, at 5pm Pacific time. The data are here:

In [ ]:

```
kickstarter = pd.read_csv("https://github.com/jkropko/DS-6001/raw/master/localdata/
kickstarter
```

Out[ ]:

	Unnamed: 0	amt.pledged	blurb	by	country	currency	end.time	locatio
0	0	15823.0	\n'Catalysts, Explorers & Secret Keepers: Wome...	Museum of Science Fiction	US	usd	2016-11-01T23:59:00-04:00	Washingto D
1	1	6859.0	\nA unique handmade picture book for kids & ar...	Tyrone Wells & Broken Eagle, LLC	US	usd	2016-11-25T01:13:33-05:00	Portlar O
2	2	17906.0	\nA horror comedy about a repairman who was in...	Tessa Stone	US	usd	2016-11-23T23:00:00-05:00	Los Angele C
3	3	67081.0	\nThe Johnny Wander autobio omnibus you've all...	Johnny Wander	US	usd	2016-11-01T23:50:00-04:00	Brookly N
4	4	32772.0	\nThe vision for this project is the establish...	Beau's All Natural Brewing Company	RW	cad	2016-11-18T23:05:48-05:00	Kiga Rwan
...	...	...	...	...	...	...	...	
3995	3995	4403.0	\nEARTH IS BUT ONE FRUIT ON THE TREE OF LIFE. ...	Lewis Brown	US	usd	2016-11-20T01:10:00-05:00	Denver, C
3996	3996	1304.0	\nImagine designing an item with an easy-to-us...	Your Expressions	US	usd	2016-11-15T16:00:00-05:00	S Francisc C
3997	3997	1.0	\nUnique themed London venue and hostel for 9g...	Martin Wojtala	GB	gbp	2016-10-30T09:36:06-04:00	London, U
3998	3998	10.0	\nAll in One Phone Case\n	All in One Phone Case	US	usd	2016-11-17T12:11:26-05:00	Tallahasse

	Unnamed: 0	amt.pledged	blurb	by	country	currency	end.time	locati
3999	3999	35.0	\nLuxury Sunglasses built with Titanium, Carbo...	Carlos Araujo	US	usd	2016-12-11T00:11:01-05:00	New Yo N

4000 rows × 13 columns

Part a

Notice that the `end.time` column, the date and time at which the project stops accepting pledges, is formatted as follows:

```
2016-11-01T23:59:00-04:00
```

This formatting is "YYYY-MM-DDThh:mm:ss-TZD": four digits for the year, a dash, two digits for the month, another dash, and two digits for the day; the "T" separates the dates from the time; two digits for the hour, minute and second, separated by colons; and the time zone expressed as hours difference from Greenwich mean time (also called UTC), and -04:00 is four hours earlier than UTC, for example.

But `end.time` is also currently read as a string, with `object` data type:

```
In [ ]: kickstarter.dtypes
```

```
Out[ ]: Unnamed: 0              int64
        amt.pledged           float64
        blurb                  object
        by                     object
        country                object
        currency               object
        end.time               object
        location               object
        percentage.funded       int64
        state                  object
        title                  object
        type                   object
        url                    object
        dtype: object
```

Convert `end.time` to a timestamp, and extract the month, day, year, hour, minute, and second of the end time. To allow the `pd.to_datetime()` function to read timezones, use the `utc=True` argument. (2 points)

```
In [ ]: df_kickstarter_a = kickstarter.copy()
        df_kickstarter_a['end.time'] = pd.to_datetime(df_kickstarter_a['end.time'], infer_d

        df_kickstarter_a['month'] = [x.month for x in df_kickstarter_a['end.time']]
```

```
df_kickstarter_a['day'] = [x.day for x in df_kickstarter_a['end.time']]
df_kickstarter_a['year'] = [x.year for x in df_kickstarter_a['end.time']]
df_kickstarter_a['hour'] = [x.hour for x in df_kickstarter_a['end.time']]
df_kickstarter_a['minute'] = [x.minute for x in df_kickstarter_a['end.time']]
df_kickstarter_a['second'] = [x.second for x in df_kickstarter_a['end.time']]

df_kickstarter_a.head()
```

Out[ ]:

	Unnamed: 0	amt.pledged	blurb	by	country	currency	end.time	location
0	0	15823.0	\n'Catalysts, Explorers & Secret Keepers: Wome...	Museum of Science Fiction	US	usd	2016-11-02 03:59:00+00:00	Washington, DC
1	1	6859.0	\nA unique handmade picture book for kids & ar...	Tyrone Wells & Broken Eagle, LLC	US	usd	2016-11-25 06:13:33+00:00	Portland, OR
2	2	17906.0	\nA horror comedy about a repairman who was in...	Tessa Stone	US	usd	2016-11-24 04:00:00+00:00	Los Angeles, CA
3	3	67081.0	\nThe Johnny Wander autobio omnibus you've all...	Johnny Wander	US	usd	2016-11-02 03:50:00+00:00	Brooklyn, NY
4	4	32772.0	\nThe vision for this project is the establish...	Beau's All Natural Brewing Company	RW	cad	2016-11-19 04:05:48+00:00	Kigali, Rwanda

In [ ]:

```
df_kickstarter_a.dtypes
```

Out[ ]:

```
Unnamed: 0                           int64
amt.pledged                        float64
blurb                               object
by                                  object
country                             object
currency                            object
end.time               datetime64[ns, UTC]
location                            object
percentage.funded                    int64
state                               object
title                               object
type                                object
url                                 object
month                                int64
day                                  int64
year                                 int64
hour                                 int64
minute                               int64
second                               int64
dtype: object
```

Part b

Create a dataframe with one row for every ending day in the `kickstarter` data that reports the average amount pledged (`amt.pledged`) on each day. Sort the rows in descending order by average amount pledged, and display the five days with the highest averages. (2 points)

In [ ]:

```
df_kickstarter_b = df_kickstarter_a.groupby(['day']).agg({'amt.pledged': 'mean'}).s
df_kickstarter_b.reset_index(inplace=True)
df_kickstarter_b.columns = ['day', 'avg_amt_pledged']
df_kickstarter_b['avg_amt_pledged'] = df_kickstarter_b['avg_amt_pledged'].round(2)
df_kickstarter_b.head(5)
```

Out[ ]:

	day	avg_amt_pledged
0	4	22503.38
1	11	22015.36
2	18	19555.57
3	5	16316.44
4	14	14540.92

Part c

Display the text of the longest `blurb` in the data. (2 points)

In [ ]:

```
df_kickstarter_c = df_kickstarter_a.copy()
df_kickstarter_c['length'] = df_kickstarter_c['blurb'].str.len() # get the length o
df_kickstarter_c.sort_values(by='length', ascending = False)[['blurb','length']].he
```

Out[ ]:

	blurb	length
2413	\nWe are charismatic anti-rock band hailing fr...	137
1686	\nWe are excited to introduce to you our new L...	137
714	\nNous souhaitons créer une Safranière. Nous p...	137
1721	\nLove sloths? Here's your chance to gear up a...	137
2834	\nBruno Charts the highs and lows of Frank Bru...	137
2844	\nI am making a documentary film about JonBene...	137
357	\nA fully illustrated Pre-Retrospective look f...	137
2849	\nTeams will dash through their towns collecti...	137
718	\nCreating memories for low income families/su...	137
1717	\nFrom the moment we flew in to the world of T...	137

Part c Answer, There are Many Blurbs that are 137 Characters, but here is the one sorted at the top:

In [ ]:

```
df_kickstarter_c.iloc[0]['blurb']
```

Out[ ]:

```
"\n'Catalysts, Explorers & Secret Keepers: Women of Science Fiction' is a take-hom
e exhibit & anthology by the Museum of Science Fiction.\n"
```

Part d

How many blurbs for projects with end dates between November 15, 2016 and December 7, 2016 contain the phrase "science fiction"? [Hint: Don't forget to make this search case-insensitive and to sort the `kickstarter` dataframe by `end.time` before setting `end.time` as the index.] (2 points)

In [ ]:

```
df_kickstarter_c.sort_values(by='end.time', ascending = False, inplace=True)
df_kickstarter_c.head()
```

Out[ ]:

	Unnamed: 0	amt.pledged	blurb	by	country	currency	end.time	loca
676	676	0.0	\nA story that follows a freelance photographe...	Raul Colon	US	usd	2016-12-28 22:26:49+00:00	Orland
2159	2159	100.0	\nAnnual CIMSEC Outreach in international mari...	CIMSEC Treasurer	US	usd	2016-12-28 18:32:05+00:00	Washing
452	452	4701.0	\nLil' Bunny Sue Roux is part cat, bunny, kang...	Golden Bell Studios	US	usd	2016-12-28 17:59:22+00:00	Orlean
3589	3589	0.0	\nA kind of Music Tale is a 360° video documen...	Maud Watel Kazak	FR	eur	2016-12-28 17:06:50+00:00	Fra Fr
3500	3500	328.0	\nReal Indian Chai Premix with added low sugar...	Khan Luxury	CZ	eur	2016-12-28 14:59:39+00:00	Pra C Rep

Create the index on `end.time` and check that the index works

In [ ]:

```
df_kickstarter_c.index = df_kickstarter_c['end.time']
df_kickstarter_c['11/15/2016':'12/07/2016'].head() # just checking that this works
```

Out[ ]:

	Unnamed: 0	amt.pledged	blurb	by	country	currency	end.tim
end.time							
2016-12-07 20:58:05+00:00	2600	0.0	\nThe Cerberus Files is a six-mission custom c...	Ian Sanderson	US	usd	2016-12-0 20:58:05+00:0
2016-12-07 20:05:20+00:00	2469	158.0	\nWe present only the multifunction wallets, m...	DA VINCI workshop	UA	usd	2016-12-0 20:05:20+00:0
2016-12-07 19:19:43+00:00	3304	1602.0	\nJoin Keyed-In 2 Christ (Emma & Kristina) as ...	Keyed-In 2 Christ	US	usd	2016-12-0 19:19:43+00:0
2016-12-07 10:59:38+00:00	3599	250.0	\nStep into an immersive experience of compass...	Sciosity	AU	aud	2016-12-0 10:59:38+00:0
2016-12-07 08:11:19+00:00	1656	16347.0	\nControl and manage any ERNEST device for bot...	ERNEST	US	usd	2016-12-0 08:11:19+00:0

Now, I need to do the search for contains the phrase "science fiction"

In [ ]:

```
map_contains_sf = df_kickstarter_c['11/15/2016':'12/07/2016']['blurb'].str.lower().
# only show the rows where the blurb contains the word 'science fiction'
df_daterange_sf = df_kickstarter_c['11/15/2016':'12/07/2016'][map_contains_sf]
print(df_daterange_sf.shape)
df_daterange_sf.head(6)
```

```
(6, 20)
```

Out[ ]:

	Unnamed: 0	amt.pledged	blurb	by	country	currency	end.time
end.time							
2016-12-07 03:25:01+00:00	604	5299.0	\nA Science Fiction film filled with entertain...	Chris	US	usd	2016-12-07 03:25:01+00:00
2016-11-30 22:00:00+00:00	2500	435.0	\nAn anthology of science fiction and fantasy ...	Cheryl Morgan	GB	gbp	2016-11-30 22:00:00+00:00
2016-11-29 01:00:00+00:00	214	5781.0	\nLegendary science fiction authors and the ma...	Randy Ritnour	US	usd	2016-11-29 01:00:00+00:00
2016-11-18 07:31:01+00:00	576	21364.0	\nSpruitje makes futuristic designs with light...	Spruitje	NL	eur	2016-11-18 07:31:01+00:00
2016-11-18 06:15:00+00:00	3386	875.0	\nThe Exodus Commission is a Christian sci-fi ...	Virtual Exodus	US	usd	2016-11-18 06:15:00+00:00
2016-11-17 19:57:17+00:00	3406	0.0	\nScience fiction Action adventure comedy tech...	LaNard Morrison	US	usd	2016-11-17 19:57:17+00:00

There are 6 records in the date range '11/15/2016':'12/07/2016' where the blurb contains the phrase 'science fiction'