In [1]:

```
"""
Please execute the code below and observe the output you get. Also, please learn how to use each of these statements
to get a similar task done.
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
df =
pd.read_csv('https://raw.githubusercontent.com/jackiekazil/data-wrangling/master/data/chp3/data-text.csv')
df.head(2)
df1 =
pd.read_csv('https://raw.githubusercontent.com/kjam/data-wrangling-pycon/master/d ata/berlin_weather_oldest.csv')
df1.head(2)

"""
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
# Reading csv file using pandas from the url 'https://raw.githubusercontent.com/jackiekazil/data-wrangling/master/data
#chp3/data-text.csv'
df = pd.read_csv('https://raw.githubusercontent.com/jackiekazil/data-wrangling/master/data/chp3/data-text.csv')
# displaying top 2 records for the file
df.head(2)
```

Out[1]:

	Indicator	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	NaN
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	NaN

In [2]:
```
# Reading csv file using pandas from the url 'https://raw.githubusercontent.com/kjam/data-wrangling-pycon/master/data/
#berlin_weather_oldest.csv'
df1 = pd.read_csv('https://raw.githubusercontent.com/kjam/data-wrangling-pycon/master/data/berlin_weather_oldest.csv'
# displaying top 2 records for the file
df1.head(2)
```

Out[2]:

	STATION	STATION_NAME	DATE	PRCP	SNWD	SNOW	TMAX	TMIN	WDFG	PGTM	...	WT09	WT07	WT01	WT06	WT05	W
0	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310101	46	-9999	-9999	-9999	-11	-9999	-9999	...	-9999	-9999	-9999	-9999	-9999	-9
1	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310102	107	-9999	-9999	50	11	-9999	-9999	...	-9999	-9999	-9999	-9999	-9999	-9

2 rows × 21 columns

1. Get the Metadata from the above files.

In [3]:
```
# to display the metadata information about the data frame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4656 entries, 0 to 4655
Data columns (total 12 columns):
Indicator                  4656 non-null object
PUBLISH STATES             4656 non-null object
Year                       4656 non-null int64
WHO region                 4656 non-null object
World Bank income group    4656 non-null object
Country                    4656 non-null object
Sex                        4656 non-null object
Display Value              4656 non-null int64
Numeric                    4656 non-null float64
Low                        0 non-null float64
High                       0 non-null float64
Comments                   0 non-null float64
dtypes: float64(4), int64(2), object(6)
memory usage: 436.6+ KB
```

In [4]:
```
# to display the metadata information about the data frame
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117208 entries, 0 to 117207
Data columns (total 21 columns):
STATION         117208 non-null object
STATION_NAME    117208 non-null object
DATE            117208 non-null int64
PRCP            117208 non-null int64
SNWD            117208 non-null int64
SNOW            117208 non-null int64
TMAX            117208 non-null int64
TMIN            117208 non-null int64
WDFG            117208 non-null int64
PGTM            117208 non-null int64
WSFG            117208 non-null int64
WT09            117208 non-null int64
WT07            117208 non-null int64
WT01            117208 non-null int64
WT06            117208 non-null int64
WT05            117208 non-null int64
WT04            117208 non-null int64
WT16            117208 non-null int64
WT08            117208 non-null int64
WT18            117208 non-null int64
WT03            117208 non-null int64
dtypes: int64(19), object(2)
memory usage: 18.8+ MB
```

2. Get the row names from the above files.

In [5]:
```
# displaying the rows which are index values of the data frame in an array
row=df.index.values
row
```

Out[5]:
```
array([   0,    1,    2, ..., 4653, 4654, 4655], dtype=int64)
```

In [6]:
```
# displaying the rows which are index values of the data frame in an array
row1=df1.index.values
row1
```

Out[6]:
```
array([     0,      1,      2, ..., 117205, 117206, 117207], dtype=int64)
```

3. Change the column name from any of the above file.

In [7]:
```
# Column Name modified from Indicator to Indicator_id
df.rename(columns={'Indicator':'Indicator_id'})
```

Out[7]:

	Indicator_id	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	NaN
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	NaN
2	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Andorra	Female	28	28.0	NaN	NaN	NaN
3	Life expectancy at age 60 (years)	Published	2000	Europe	High-income	Andorra	Both sexes	23	23.0	NaN	NaN	NaN
4	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	United Arab Emirates	Female	78	78.0	NaN	NaN	NaN
5	Life expectancy at birth (years)	Published	2000	Americas	High-income	Antigua and Barbuda	Male	72	72.0	NaN	NaN	NaN
6	Life expectancy at age	Published	1990	Americas	High-income	Antigua and	Male	17	17.0	NaN	NaN	NaN

In [8]:
```
# column name not modified permanantly with above code
df.head(2)
```

Out[8]:

	Indicator	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	NaN
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	NaN

4. Change the column name from any of the above file and store the changes made permanently.

In [9]:
```

# Column Name modified from Indicator to Indicator_id permanantly

df.rename(columns={'Indicator':'Indicator_id'}, inplace=True)
df.head(2)

```

Out[9]:

	Indicator_id	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	NaN
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	NaN

```

Another way to modify the column name permanantly is

Get ndArray of all column names : columnsNames = df.columns.values
Modify the first column name : columnsNames[0] = 'Indicator_id'

```

5. Change the names of multiple columns.

In [10]:
```
# Column Name modified from PUBLISH STATES to Publication StatusIndicator_id  and WHO region to WHO Regionpermanantly
df.rename(columns={'PUBLISH STATES':'Publication Status','WHO region':'WHO Region'}, inplace=True)
df.head(2)
```

Out[10]:

	Indicator_id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	NaN
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	NaN

6. Arrange values of a particular column in ascending order.

In [11]:
```
# sorting the dataframe df by ascending order based on column Year
#(default value for ascending is true so by default the soring will be ascending order)
df.sort_values(by=['Year'])
```

Out[11]:

	Indicator_id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	NaN
1270	Life expectancy at birth (years)	Published	1990	Europe	High-income	Germany	Male	72	72.0	NaN	NaN	NaN
3193	Life expectancy at birth (years)	Published	1990	Europe	Lower-middle-income	Republic of Moldova	Male	65	65.0	NaN	NaN	NaN
3194	Life expectancy at birth (years)	Published	1990	Europe	Lower-middle-income	Republic of Moldova	Both sexes	68	68.0	NaN	NaN	NaN
3197	Life expectancy at age 60 (years)	Published	1990	Europe	Lower-middle-income	Republic of Moldova	Male	15	15.0	NaN	NaN	NaN
1264	Life expectancy at	Published	1990	Europe	High-income	Cyprus	Both	76	76.0	NaN	NaN	NaN

7. Arrange multiple column values in ascending order.

In [12]:
```
# new dataframe created with the below column values
newdf=df.loc[:,['Indicator_id','Country','Year','WHO Region','Publication Status']]
```

In [13]:
```
#soring the values based on multiple column values with ascending
#(default value for ascending is true so by default the soring will be ascending order)
newdf.sort_values(by=['Country','Year'])
newdf.head(4)
```

Out[13]:

	Indicator_id	Country	Year	WHO Region	Publication Status
0	Life expectancy at birth (years)	Andorra	1990	Europe	Published
1	Life expectancy at birth (years)	Andorra	2000	Europe	Published
2	Life expectancy at age 60 (years)	Andorra	2012	Europe	Published
3	Life expectancy at age 60 (years)	Andorra	2000	Europe	Published

8. Make countryas the first column of the dataframe.

In [14]:
```
#get a list of columns
cols = list(df)
# move the column to head of list using index, pop and insert
cols.insert(0, cols.pop(cols.index('Country')))
# using loc to reorder the columns
df_update = df.loc[:, cols]
df_update.head(4)
```

Out[14]:

	Country	Indicator_id	Publication Status	Year	WHO Region	World Bank income group	Sex	Display Value	Numeric	Low	High	Comments
0	Andorra	Life expectancy at birth (years)	Published	1990	Europe	High-income	Both sexes	77	77.0	NaN	NaN	NaN
1	Andorra	Life expectancy at birth (years)	Published	2000	Europe	High-income	Both sexes	80	80.0	NaN	NaN	NaN
2	Andorra	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Female	28	28.0	NaN	NaN	NaN
3	Andorra	Life expectancy at age 60 (years)	Published	2000	Europe	High-income	Both sexes	23	23.0	NaN	NaN	NaN

9. Get the column array using a variable

In [15]:
```
# Selecting the WHO region column values into one array variable
col2=df['WHO Region'].values
#Displaying vallues of the array variable
col2
```

Out[15]:
```
array(['Europe', 'Europe', 'Europe', ..., 'Africa', 'Africa', 'Africa'],
      dtype=object)
```

10. Get the subset rows 11, 24, 37

In [16]:
```
# Selecting the specified rows with all the column values into one data set
subset=df.iloc[[11,24,37],:]
#displaying the subset dataframe
subset
```

Out[16]:

	Indicator_id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
11	Life expectancy at birth (years)	Published	2012	Europe	High-income	Austria	Female	83	83.0	NaN	NaN	NaN
24	Life expectancy at age 60 (years)	Published	2012	Western Pacific	High-income	Brunei Darussalam	Female	21	21.0	NaN	NaN	NaN
37	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Cyprus	Female	26	26.0	NaN	NaN	NaN

11. Get the subset rows excluding 5, 12, 23, and 56

In [17]:
```
# Dropping the rows 5,12,23 and 56 using drop by row index vlues
df.drop([5,12,23,56]).head(10) # row index 5 is missed in below output of first 10 rows
```

Out[17]:

	Indicator_id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	NaN	NaN	NaN
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	NaN	NaN	NaN
2	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Andorra	Female	28	28.0	NaN	NaN	NaN
3	Life expectancy at age 60 (years)	Published	2000	Europe	High-income	Andorra	Both sexes	23	23.0	NaN	NaN	NaN
4	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	United Arab Emirates	Female	78	78.0	NaN	NaN	NaN
6	Life expectancy at age 60 (years)	Published	1990	Americas	High-income	Antigua and Barbuda	Male	17	17.0	NaN	NaN	NaN
7	Life expectancy at age 60 (years)	Published	2012	Americas	High-income	Antigua and Barbuda	Both sexes	22	22.0	NaN	NaN	NaN
8	Life expectancy at birth (years)	Published	2012	Western Pacific	High-income	Australia	Male	81	81.0	NaN	NaN	NaN
9	Life expectancy at birth (years)	Published	2000	Western Pacific	High-income	Australia	Both sexes	80	80.0	NaN	NaN	NaN
10	Life expectancy at birth (years)	Published	2012	Western Pacific	High-income	Australia	Both sexes	83	83.0	NaN	NaN	NaN

Load datasets from CSV

In [18]:
```
users = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/users.csv')
sessions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/sessions.csv')
products = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/products.csv')
transactions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/transactions.csv')
```

In [19]:
```
users.head(2)
```

Out[19]:

	UserID	User	Gender	Registered	Cancelled
0	1	Charles	male	2012-12-21	NaN
1	2	Pedro	male	2010-08-01	2010-08-08

In [20]:
```
sessions.head(2)
```

Out[20]:

	SessionID	SessionDate	UserID
0	1	2010-01-05	2
1	2	2010-08-01	2

In [21]:
```
products.head(2)
```

Out[21]:

	ProductID	Product	Price
0	1	A	14.16
1	2	B	33.04

In [22]:
```
transactions.head(2)
```

Out[22]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity
0	1	2010-08-21	7.0	2	1
1	2	2011-05-26	3.0	4	1

12. Join users to transactions, keeping all rows from transactions and only matching rows from users (left join)

In [23]:
```
#Merging the data frame using Left join based on user ID and assigning to result dataframe
result = pd.merge(transactions, users, on='UserID', how='left', sort=False);
#Displaying the result data fram
result
```

Out[23]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Cancelled
0	1	2010-08-21	7.0	2	1	NaN	NaN	NaN	NaN
1	2	2011-05-26	3.0	4	1	Caroline	female	2012-10-23	2016-06-07
2	3	2011-06-16	3.0	3	1	Caroline	female	2012-10-23	2016-06-07
3	4	2012-08-26	1.0	2	3	Charles	male	2012-12-21	NaN
4	5	2013-06-06	2.0	4	1	Pedro	male	2010-08-01	2010-08-08
5	6	2013-12-23	2.0	5	6	Pedro	male	2010-08-01	2010-08-08
6	7	2013-12-30	3.0	4	1	Caroline	female	2012-10-23	2016-06-07
7	8	2014-04-24	NaN	2	3	NaN	NaN	NaN	NaN
8	9	2015-04-24	7.0	4	3	NaN	NaN	NaN	NaN
9	10	2016-05-08	3.0	4	4	Caroline	female	2012-10-23	2016-06-07

13. Which transactions have a UserID not in users?

In [24]:
```
# Merging two data frames with lef join as transactions table as left with indicator as True
Merged = pd.merge(transactions, users, on='UserID', how='left', sort=False,indicator=True);
# filtering only left only rows which results rows which are in only transaction dataframe (like Outer - Inner join)
result1=Merged[Merged['_merge']=='left_only']
#Displaying only Transaction data frame columns
result1.iloc[:,0:5]
```

Out[24]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity
0	1	2010-08-21	7.0	2	1
7	8	2014-04-24	NaN	2	3
8	9	2015-04-24	7.0	4	3

14. Join users to transactions, keeping only rows from transactions and users that match via UserID (inner join)

In [25]:
```
# Joining the two data frames with inner join on UserID column and assigning to the result2
result2 = pd.merge(transactions, users, on='UserID', how='inner', sort=False);
#Displaying the result data fram
result2
```

Out[25]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Cancelled
0	2	2011-05-26	3.0	4	1	Caroline	female	2012-10-23	2016-06-07
1	3	2011-06-16	3.0	3	1	Caroline	female	2012-10-23	2016-06-07
2	7	2013-12-30	3.0	4	1	Caroline	female	2012-10-23	2016-06-07
3	10	2016-05-08	3.0	4	4	Caroline	female	2012-10-23	2016-06-07
4	4	2012-08-26	1.0	2	3	Charles	male	2012-12-21	NaN
5	5	2013-06-06	2.0	4	1	Pedro	male	2010-08-01	2010-08-08
6	6	2013-12-23	2.0	5	6	Pedro	male	2010-08-01	2010-08-08

15. Join users to transactions, displaying all matching rows AND all non-matching rows (full outer join)

In [26]:
```
# Joining the two data frames with Outer join on UserID column and assigning to the result3
result3 = pd.merge(transactions, users, on='UserID', how='outer', sort=False);
#Displaying the result data fram
result3
```

Out[26]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Cancelled
0	1.0	2010-08-21	7.0	2.0	1.0	NaN	NaN	NaN	NaN
1	9.0	2015-04-24	7.0	4.0	3.0	NaN	NaN	NaN	NaN
2	2.0	2011-05-26	3.0	4.0	1.0	Caroline	female	2012-10-23	2016-06-07
3	3.0	2011-06-16	3.0	3.0	1.0	Caroline	female	2012-10-23	2016-06-07
4	7.0	2013-12-30	3.0	4.0	1.0	Caroline	female	2012-10-23	2016-06-07
5	10.0	2016-05-08	3.0	4.0	4.0	Caroline	female	2012-10-23	2016-06-07
6	4.0	2012-08-26	1.0	2.0	3.0	Charles	male	2012-12-21	NaN
7	5.0	2013-06-06	2.0	4.0	1.0	Pedro	male	2010-08-01	2010-08-08
8	6.0	2013-12-23	2.0	5.0	6.0	Pedro	male	2010-08-01	2010-08-08
9	8.0	2014-04-24	NaN	2.0	3.0	NaN	NaN	NaN	NaN
10	NaN	NaN	4.0	NaN	NaN	Brielle	female	2013-07-17	NaN
11	NaN	NaN	5.0	NaN	NaN	Benjamin	male	2010-11-25	NaN

16. Determine which sessions occurred on the same day each user registered

In [27]:
```
# Merging two data frames in to result4 where the User id registered date and session date are same
result4=pd.merge(users,sessions,left_on=[users.UserID,users.Registered],right_on=[sessions.UserID,sessions.SessionDate
#Displaying the result data frame
result4.iloc[:,2:-1]
```

Out[27]:

	UserID_x	User	Gender	Registered	Cancelled	SessionID	SessionDate
0	2	Pedro	male	2010-08-01	2010-08-08	2	2010-08-01
1	4	Brielle	female	2013-07-17	NaN	9	2013-07-17

17. Build a dataset with every possible (UserID, ProductID) pair (cross join)

In [28]:
```
# Creating Tempaorary column  with value 1 in both the data frames for join
users['temp']=1
products['temp']=1
# Merging dataframes into results based on Temp column with Outer Join
result5=pd.merge(users,products,on='temp',how='outer')
#Displaying the required columns UserID and ProductID from result data frame
result5.loc[:,["UserID","ProductID"]]
```

Out[28]:

	UserID	ProductID
0	1	1
1	1	2
2	1	3
3	1	4
4	1	5
5	2	1
6	2	2
7	2	3
8	2	4
9	2	5
10	3	1
11	3	2
12	3	3
13	3	4
14	3	5
15	4	1
16	4	2
17	4	3
18	4	4
19	4	5

	UserID	ProductID
20	5	1
21	5	2
22	5	3
23	5	4
24	5	5

In [ ]:
```
1
```