DSM Project 1

In [1]:
```
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)
```

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]:
```
df1 = pd.read_csv('https://raw.githubusercontent.com/kjam/data-wrangling-pycon/master/data/berlin_weather_oldest.csv')
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

In [3]:
```
# 1. Get the Metadata from the above files.
# Solution
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]:
```
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
```

In [5]:
```
#2. Get the row names from the above files.
# Solution
np.array(df.index)
```

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

In [6]:
```
np.array(df1.index)
```

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

In [7]:
```
# 3. Change the column name from any of the above file.
# Solution
df_temp = df.rename(columns={'Indicator': 'Indicator_id'})
df_temp.head(2)
```

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

In [8]:
```
#4.Change the column name from any of the above file and store the changes made permanently.
df.rename(columns={'Indicator': 'Indicator_id'},inplace=True)
df.head(2)
```

Out[8]:

	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

In [9]:
```
# 5. Change the names of multiple columns.
df.rename(columns={'PUBLISH STATES': 'Publication Status','WHO region':'WHO Region'},inplace=True)
df.head(2)
```

Out[9]:

	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

In [10]:
```
# 6. Arrange values of a particular column in ascending order.
df.sort_values('Year').head()
```

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
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

In [11]:
```
#7. Arrange multiple column values in ascending order.
df.sort_values(['Indicator_id','Country','Year','WHO Region','Publication Status'])
df[['Indicator_id','Country','Year','WHO Region','Publication Status']].head(3)
```

Out[11]:

	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

In [12]:
```
# 8. Make country as the first column of the dataframe.
df[['Country', 'Indicator_id', 'Publication Status', 'Year','WHO Region','World Bank income group','Sex','Display Value',
```

Out[12]:

	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
4	United Arab Emirates	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	Female	78	78.0	NaN	NaN	NaN

In [13]:
```
#9. Get the column array using a variable
col = np.array(df['WHO Region'])
col
```

Out[13]:
```
array(['Europe', 'Europe', 'Europe', ..., 'Africa', 'Africa', 'Africa'],
      dtype=object)
```

In [14]:
```
# 10. Get the subset rows 11, 24, 37
df.loc[[11,24,37]]
```

Out[14]:

	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

In [15]:
```
# 11. Get the subset rows excluding 5, 12, 23, and 56
df_exclude = df.index.isin([3,5])
df[~df_exclude].head()
```

Out[15]:

	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
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

In [16]:
```
# Loading users csv file
users = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/users.csv' )
users.head()
```

Out[16]:

	UserID	User	Gender	Registered	Cancelled
0	1	Charles	male	2012-12-21	NaN
1	2	Pedro	male	2010-08-01	2010-08-08
2	3	Caroline	female	2012-10-23	2016-06-07
3	4	Brielle	female	2013-07-17	NaN
4	5	Benjamin	male	2010-11-25	NaN

In [17]:
```
sessions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/sessions.csv')
sessions.head()
```

Out[17]:

	SessionID	SessionDate	UserID
0	1	2010-01-05	2
1	2	2010-08-01	2
2	3	2010-11-25	2
3	4	2011-09-21	5
4	5	2011-10-19	4

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

Out[18]:

	ProductID	Product	Price
0	1	A	14.16
1	2	B	33.04
2	3	C	10.65
3	4	D	10.02
4	5	E	29.66

In [19]:
```
transactions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/transactions.csv')
transactions
```

Out[19]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity
0	1	2010-08-21	7.0	2	1
1	2	2011-05-26	3.0	4	1
2	3	2011-06-16	3.0	3	1
3	4	2012-08-26	1.0	2	3
4	5	2013-06-06	2.0	4	1
5	6	2013-12-23	2.0	5	6
6	7	2013-12-30	3.0	4	1
7	8	2014-04-24	NaN	2	3
8	9	2015-04-24	7.0	4	3
9	10	2016-05-08	3.0	4	4

In [37]:
```
# 12. Join users to transactions, keeping all rows from transactions and only matching rows from users (left join)
pd.merge(transactions,users , on='UserID', how='left').fillna({'Registered':pd.NaT,'Cancelled':pd.NaT})
```

Out[37]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Cancelled
0	1	2010-08-21	7.0	2	1	NaN	NaN	NaT	NaT
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	NaT
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	NaT	NaT
8	9	2015-04-24	7.0	4	3	NaN	NaN	NaT	NaT
9	10	2016-05-08	3.0	4	4	Caroline	female	2012-10-23	2016-06-07

In [22]:
```
# 13. Which transactions have a UserID not in users?
mergedDF = transactions.merge(users,on=['UserID'])
transactions[(~transactions.UserID.isin(mergedDF.UserID))]
```

Out[22]:

	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

In [38]:
```
# 14. Join users to transactions, keeping only rows from transactions and users that match via UserID (inner join)
pd.merge(transactions,users, on='UserID').fillna({'Cancelled':pd.NaT})
```

Out[38]:

	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	NaT
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

In [39]:
```
# 15. Join users to transactions, displaying all matching rows AND all non-matching rows (full outer join)
pd.merge(transactions, users, on='UserID', how='outer').fillna({'Registered':pd.NaT,'Cancelled':pd.NaT,'TransactionDate':
```

Out[39]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Cancelled
0	1.0	2010-08-21	7.0	2.0	1.0	NaN	NaN	NaT	NaT
1	9.0	2015-04-24	7.0	4.0	3.0	NaN	NaN	NaT	NaT
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	NaT
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	NaT	NaT
10	NaN	NaT	4.0	NaN	NaN	Brielle	female	2013-07-17	NaT
11	NaN	NaT	5.0	NaN	NaN	Benjamin	male	2010-11-25	NaT

In [40]:
```
# 16. Determine which sessions occurred on the same day each user registered
pd.merge(users, sessions, right_on=['SessionDate','UserID'],left_on=['Registered','UserID'])
```

Out[40]:

	UserID	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

In [44]:
```
#17. Build a dataset with every possible (UserID, ProductID) pair (cross join)
users.assign(foo=1).merge(products.assign(foo=1)).drop('foo', 1)[['UserID','ProductID']]
```

Out[44]:

	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
20	5	1
21	5	2
22	5	3

	UserID	ProductID
23	5	4
24	5	5

In [45]:
```
# 18. Determine how much quantity of each product was purchased by each user
df = users.assign(foo=1).merge(products.assign(foo=1)).drop('foo', 1)
df = pd.merge(df[['UserID','ProductID']] ,transactions, on=['UserID','ProductID'], how='left').fillna(0)
df[['UserID','ProductID','Quantity']]
```

Out[45]:

	UserID	ProductID	Quantity
0	1	1	0.0
1	1	2	3.0
2	1	3	0.0
3	1	4	0.0
4	1	5	0.0
5	2	1	0.0
6	2	2	0.0
7	2	3	0.0
8	2	4	1.0
9	2	5	6.0
10	3	1	0.0
11	3	2	0.0
12	3	3	1.0
13	3	4	1.0
14	3	4	1.0
15	3	4	4.0
16	3	5	0.0
17	4	1	0.0
18	4	2	0.0
19	4	3	0.0
20	4	4	0.0
21	4	5	0.0

	UserID	ProductID	Quantity
22	5	1	0.0
23	5	2	0.0
24	5	3	0.0
25	5	4	0.0
26	5	5	0.0

In [46]:
```
# 19. For each user, get each possible pair of pair transactions (TransactionID1,TransacationID2)
pd.merge(transactions,transactions, on="UserID")
```

Out[46]:

	TransactionID_x	TransactionDate_x	UserID	ProductID_x	Quantity_x	TransactionID_y	TransactionDate_y	ProductID_y	Quantity_y
0	1	2010-08-21	7.0	2	1	1	2010-08-21	2	1
1	1	2010-08-21	7.0	2	1	9	2015-04-24	4	3
2	9	2015-04-24	7.0	4	3	1	2010-08-21	2	1
3	9	2015-04-24	7.0	4	3	9	2015-04-24	4	3
4	2	2011-05-26	3.0	4	1	2	2011-05-26	4	1
5	2	2011-05-26	3.0	4	1	3	2011-06-16	3	1
6	2	2011-05-26	3.0	4	1	7	2013-12-30	4	1
7	2	2011-05-26	3.0	4	1	10	2016-05-08	4	4
8	3	2011-06-16	3.0	3	1	2	2011-05-26	4	1
9	3	2011-06-16	3.0	3	1	3	2011-06-16	3	1
10	3	2011-06-16	3.0	3	1	7	2013-12-30	4	1
11	3	2011-06-16	3.0	3	1	10	2016-05-08	4	4
12	7	2013-12-30	3.0	4	1	2	2011-05-26	4	1
13	7	2013-12-30	3.0	4	1	3	2011-06-16	3	1
14	7	2013-12-30	3.0	4	1	7	2013-12-30	4	1
15	7	2013-12-30	3.0	4	1	10	2016-05-08	4	4
16	10	2016-05-08	3.0	4	4	2	2011-05-26	4	1
17	10	2016-05-08	3.0	4	4	3	2011-06-16	3	1
18	10	2016-05-08	3.0	4	4	7	2013-12-30	4	1
19	10	2016-05-08	3.0	4	4	10	2016-05-08	4	4
20	4	2012-08-26	1.0	2	3	4	2012-08-26	2	3
21	5	2013-06-06	2.0	4	1	5	2013-06-06	4	1
22	5	2013-06-06	2.0	4	1	6	2013-12-23	5	6

	TransactionID_x	TransactionDate_x	UserID	ProductID_x	Quantity_x	TransactionID_y	TransactionDate_y	ProductID_y	Quantity_y
23	6	2013-12-23	2.0	5	6	5	2013-06-06	4	1
24	6	2013-12-23	2.0	5	6	6	2013-12-23	5	6
25	8	2014-04-24	NaN	2	3	8	2014-04-24	2	3

In [48]:
```
# 20. Join each user to his/her first occuring transaction in the transactions table
users.merge(transactions.drop_duplicates('UserID'),how='left',on='UserID').fillna({'Cancelled':pd.NaT,'TransactionDate':p
```

Out[48]:

	UserID	User	Gender	Registered	Cancelled	TransactionID	TransactionDate	ProductID	Quantity
0	1	Charles	male	2012-12-21	NaT	4.0	2012-08-26	2.0	3.0
1	2	Pedro	male	2010-08-01	2010-08-08	5.0	2013-06-06	4.0	1.0
2	3	Caroline	female	2012-10-23	2016-06-07	2.0	2011-05-26	4.0	1.0
3	4	Brielle	female	2013-07-17	NaT	NaN	NaT	NaN	NaN
4	5	Benjamin	male	2010-11-25	NaT	NaN	NaT	NaN	NaN

In [49]:
```
# 21. Test to see if we can drop columns
data = users.merge(transactions.drop_duplicates('UserID'),how='left',on='UserID')
my_columns = list(data.columns)
my_columns
```

Out[49]:
```
['UserID',
 'User',
 'Gender',
 'Registered',
 'Cancelled',
 'TransactionID',
 'TransactionDate',
 'ProductID',
 'Quantity']
```

In [50]:
```
# set threshold to drop NAs
list(data.dropna(thresh=int(data.shape[0] * .9), axis=1).columns)
```

Out[50]:
```
['UserID', 'User', 'Gender', 'Registered']
```

In [51]:
```
missing_info = list(data.columns[data.isnull().any()])
missing_info
```

Out[51]:
```
['Cancelled', 'TransactionID', 'TransactionDate', 'ProductID', 'Quantity']
```

In [52]:
```
for col in missing_info:
    num_missing = data[data[col].isnull() == True].shape[0]
    print('number missing for column {}: {}'.format(col, num_missing))
```

```
number missing for column Cancelled: 3
number missing for column TransactionID: 2
number missing for column TransactionDate: 2
number missing for column ProductID: 2
number missing for column Quantity: 2
```

In [53]:
```
for col in missing_info:
    percent_missing = data[data[col].isnull() == True].shape[0] / data.shape[0]
    print('percent missing for column {}: {}'.format( col,percent_missing))
```

```
percent missing for column Cancelled: 0.6
percent missing for column TransactionID: 0.4
percent missing for column TransactionDate: 0.4
percent missing for column ProductID: 0.4
percent missing for column Quantity: 0.4
```