Untitled26

February 10, 2019

In [18]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
df = pd.read_csv('https://raw.githubusercontent.com/jackiekazil/data-wrangling/master/
df.head(2)
```

Out[18]:
```
                          Indicator PUBLISH STATES  Year WHO region  \
0  Life expectancy at birth (years)      Published  1990     Europe
1  Life expectancy at birth (years)      Published  2000     Europe

  World Bank income group  Country         Sex  Display Value  Numeric  Low  \
0             High-income  Andorra  Both sexes             77     77.0  NaN
1             High-income  Andorra  Both sexes             80     80.0  NaN

   High  Comments
0   NaN       NaN
1   NaN       NaN
```

In [2]:
```
df1 = pd.read_csv('https://raw.githubusercontent.com/kjam/data-wrangling-pycon/master/
df1.head(2)
```

Out[2]:
```
              STATION         STATION_NAME      DATE  PRCP  SNWD  SNOW  TMAX  \
0  GHCND:GME00111445  BERLIN TEMPELHOF GM  19310101    46 -9999 -9999 -9999
1  GHCND:GME00111445  BERLIN TEMPELHOF GM  19310102   107 -9999 -9999    50

   TMIN  WDFG  PGTM  ...   WT09  WT07  WT01  WT06  WT05  WT04  WT16  WT08  \
0   -11 -9999 -9999  ...  -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999
1    11 -9999 -9999  ...  -9999 -9999 -9999 -9999 -9999 -9999 -9999 -9999

   WT18  WT03
0 -9999 -9999
1 -9999 -9999

[2 rows x 21 columns]
```

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

In [3]:
```
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 [ ]: #2. Get the row names from the above files.

In [5]: rows = np.array(df.index.get_values())
        rows

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

In [6]: rows1 = np.array(df1.index.get_values())
        rows1

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

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

Out[7]:                       Indicator_id PUBLISH STATES  Year WHO region  \
        0  Life expectancy at birth (years)      Published  1990     Europe
        1  Life expectancy at birth (years)      Published  2000     Europe

          World Bank income group  Country         Sex  Display Value  Numeric  Low  \
        0             High-income  Andorra  Both sexes             77     77.0  NaN
        1             High-income  Andorra  Both sexes             80     80.0  NaN

           High  Comments
        0   NaN       NaN
        1   NaN       NaN

In [8]: df.head(2)

Out[8]:                          Indicator PUBLISH STATES  Year WHO region  \
        0  Life expectancy at birth (years)      Published  1990     Europe
        1  Life expectancy at birth (years)      Published  2000     Europe

          World Bank income group  Country         Sex  Display Value  Numeric  Low  \
        0             High-income  Andorra  Both sexes             77     77.0  NaN
        1             High-income  Andorra  Both sexes             80     80.0  NaN

           High  Comments
        0   NaN       NaN
        1   NaN       NaN

In [9]: df.rename(columns={'Indicator':'Indicator_id'},inplace=True)
        df.head(2)

Out[9]:                       Indicator_id PUBLISH STATES  Year WHO region  \
        0  Life expectancy at birth (years)      Published  1990     Europe
        1  Life expectancy at birth (years)      Published  2000     Europe

          World Bank income group  Country         Sex  Display Value  Numeric  Low  \
```

```
0             High-income  Andorra  Both sexes             77     77.0  NaN
1             High-income  Andorra  Both sexes             80     80.0  NaN

   High  Comments
0   NaN       NaN
1   NaN       NaN
```

In [10]:
```
#5. Change the names of multiple columns.

df.rename(columns={'PUBLISH STATES':'Publication Status','WHO region':'WHO Region'},in
df.head(2)
```

Out[10]:
```
                      Indicator_id Publication Status  Year WHO Region  \
0  Life expectancy at birth (years)          Published  1990     Europe
1  Life expectancy at birth (years)          Published  2000     Europe

  World Bank income group  Country         Sex  Display Value  Numeric  Low  \
0             High-income  Andorra  Both sexes             77     77.0  NaN
1             High-income  Andorra  Both sexes             80     80.0  NaN

   High  Comments
0   NaN       NaN
1   NaN       NaN
```

In [11]:
```
#6. Arrange values of a particular column in ascending order.

df.sort_values(by=['Year']).head(5)
```

Out[11]:
```
                           Indicator_id Publication Status  Year WHO Region  \
0      Life expectancy at birth (years)          Published  1990     Europe
1270   Life expectancy at birth (years)          Published  1990     Europe
3193   Life expectancy at birth (years)          Published  1990     Europe
3194   Life expectancy at birth (years)          Published  1990     Europe
3197  Life expectancy at age 60 (years)          Published  1990     Europe

     World Bank income group              Country         Sex  Display Value  \
0                High-income              Andorra  Both sexes             77
1270             High-income              Germany        Male             72
3193  Lower-middle-income  Republic of Moldova        Male             65
3194  Lower-middle-income  Republic of Moldova  Both sexes             68
3197  Lower-middle-income  Republic of Moldova        Male             15

      Numeric  Low  High  Comments
0        77.0  NaN   NaN       NaN
1270     72.0  NaN   NaN       NaN
3193     65.0  NaN   NaN       NaN
3194     68.0  NaN   NaN       NaN
3197     15.0  NaN   NaN       NaN
```

In [12]:
```
#7. Arrange multiple column values in ascending order.

df.sort_values(by=['Country','Year','WHO Region','Publication Status'],inplace=True)
#8. Make country as the first column of the dataframe.df.head(5)
```

Out[12]:

	Indicator_id	Publication Status	Year	\
554	Life expectancy at birth (years)	Published	1990	
555	Life expectancy at age 60 (years)	Published	1990	
965	Life expectancy at birth (years)	Published	1990	
1395	Life expectancy at age 60 (years)	Published	1990	
1792	Life expectancy at birth (years)	Published	1990	

	WHO Region	World Bank income group	Country	Sex	\
554	Eastern Mediterranean	Low-income	Afghanistan	Both sexes	
555	Eastern Mediterranean	Low-income	Afghanistan	Female	
965	Eastern Mediterranean	Low-income	Afghanistan	Male	
1395	Eastern Mediterranean	Low-income	Afghanistan	Both sexes	
1792	Eastern Mediterranean	Low-income	Afghanistan	Female	

	Display Value	Numeric	Low	High	Comments
554	49	49.0	NaN	NaN	NaN
555	15	15.0	NaN	NaN	NaN
965	49	49.0	NaN	NaN	NaN
1395	14	14.0	NaN	NaN	NaN
1792	50	50.0	NaN	NaN	NaN

In [13]:
```
#8. Make country as the first column of the dataframe.

df=df.reindex(columns=['Country']+[a for a in df.columns if a!='Country'])
df.head(2)
```

Out[13]:

	Country	Indicator_id	Publication Status	Year	\
554	Afghanistan	Life expectancy at birth (years)	Published	1990	
555	Afghanistan	Life expectancy at age 60 (years)	Published	1990	

	WHO Region	World Bank income group	Sex	Display Value	\
554	Eastern Mediterranean	Low-income	Both sexes	49	
555	Eastern Mediterranean	Low-income	Female	15	

	Numeric	Low	High	Comments
554	49.0	NaN	NaN	NaN
555	15.0	NaN	NaN	NaN

In [14]:
```
#9. Get the column array using a variable

df['Country'].values
```

Out[14]:
```
array(['Afghanistan', 'Afghanistan', 'Afghanistan', ..., 'Zimbabwe',
       'Zimbabwe', 'Zimbabwe'], dtype=object)
```

In [15]:
```
#10. Get the subset rows 11, 24, 37

df_temp=df.loc[[11,24,37],:]
df_temp
```

Out[15]:

	Country	Indicator_id	Publication Status	\
11	Austria	Life expectancy at birth (years)	Published	
24	Brunei Darussalam	Life expectancy at age 60 (years)	Published	
37	Cyprus	Life expectancy at age 60 (years)	Published	

	Year	WHO Region	World Bank income group	Sex	Display Value	\
11	2012	Europe	High-income	Female	83	
24	2012	Western Pacific	High-income	Female	21	
37	2012	Europe	High-income	Female	26	

	Numeric	Low	High	Comments
11	83.0	NaN	NaN	NaN
24	21.0	NaN	NaN	NaN
37	26.0	NaN	NaN	NaN

In [17]:
```
#11. Get the subset rows excluding 5, 12, 23, and 56

df_temp1=df.drop(df.index[[5,12,23,56]])
df_temp1
```

Out[17]:

	Country	Indicator_id	Publication Status	Year	\
554	Afghanistan	Life expectancy at birth (years)	Published	1990	
555	Afghanistan	Life expectancy at age 60 (years)	Published	1990	

	WHO Region	World Bank income group	Sex	Display Value	\
554	Eastern Mediterranean	Low-income	Both sexes	49	
555	Eastern Mediterranean	Low-income	Female	15	

	Numeric	Low	High	Comments
554	49.0	NaN	NaN	NaN
555	15.0	NaN	NaN	NaN

In [28]:
```
users=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/
sessions=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Da
products=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Da
transactions=pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/maste
users
```

Out[28]:

	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 [29]:
```
sessions
```

Out[29]:

	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
5	6	2012-10-23	4
6	7	2012-12-21	3
7	8	2013-05-22	4
8	9	2013-07-17	4
9	10	2016-01-11	4

In [27]:
```
transactions
```

Out[27]:

	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 [25]:
```
#12. Join users to transactions, keeping all rows from transactions and only matching
#from users (left join)

s1=pd.merge(users,transactions,how='left')
s1.head()
```

Out[25]:

	UserID	User	Gender	Registered	Cancelled	TransactionID	\
0	1	Charles	male	2012-12-21	NaN	4.0	
1	2	Pedro	male	2010-08-01	2010-08-08	5.0	
2	2	Pedro	male	2010-08-01	2010-08-08	6.0	
3	3	Caroline	female	2012-10-23	2016-06-07	2.0	
4	3	Caroline	female	2012-10-23	2016-06-07	3.0	

	TransactionDate	ProductID	Quantity
0	2012-08-26	2.0	3.0
1	2013-06-06	4.0	1.0
2	2013-12-23	5.0	6.0
3	2011-05-26	4.0	1.0
4	2011-06-16	3.0	1.0

In [26]:
```
s2=pd.merge(transactions,users,how='left')
s2
```

Out[26]:

	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

In [30]:
```
#13. Which transactions have a UserID not in users?

df3=transactions[s2['UserID'].isin(users['UserID'])==False]
df3
```

Out[30]:

	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 [31]:
```
#14. Join users to transactions, keeping only rows from transactions and users that m
#via UserID (inner join)

df4=pd.merge(transactions,users,how='inner',on='UserID')
df4
```

Out[31]:

	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

```
   Gender  Registered   Cancelled
0  female  2012-10-23  2016-06-07
1  female  2012-10-23  2016-06-07
2  female  2012-10-23  2016-06-07
3  female  2012-10-23  2016-06-07
4    male  2012-12-21         NaN
5    male  2010-08-01  2010-08-08
6    male  2010-08-01  2010-08-08
```

In [32]:
```
#15. Join users to transactions, displaying all matching rows AND all non-matching ro
#(full outer join)

df5=pd.merge(transactions,users,how='outer')
df5
```

Out[32]:
```
    TransactionID TransactionDate  UserID  ProductID  Quantity      User  \
0             1.0      2010-08-21     7.0        2.0       1.0       NaN
1             9.0      2015-04-24     7.0        4.0       3.0       NaN
2             2.0      2011-05-26     3.0        4.0       1.0  Caroline
3             3.0      2011-06-16     3.0        3.0       1.0  Caroline
4             7.0      2013-12-30     3.0        4.0       1.0  Caroline
5            10.0      2016-05-08     3.0        4.0       4.0  Caroline
6             4.0      2012-08-26     1.0        2.0       3.0   Charles
7             5.0      2013-06-06     2.0        4.0       1.0     Pedro
8             6.0      2013-12-23     2.0        5.0       6.0     Pedro
9             8.0      2014-04-24     NaN        2.0       3.0       NaN
10            NaN             NaN     4.0        NaN       NaN   Brielle
11            NaN             NaN     5.0        NaN       NaN  Benjamin

    Gender  Registered   Cancelled
0      NaN         NaN         NaN
1      NaN         NaN         NaN
2   female  2012-10-23  2016-06-07
3   female  2012-10-23  2016-06-07
4   female  2012-10-23  2016-06-07
5   female  2012-10-23  2016-06-07
6     male  2012-12-21         NaN
7     male  2010-08-01  2010-08-08
8     male  2010-08-01  2010-08-08
9      NaN         NaN         NaN
10  female  2013-07-17         NaN
11    male  2010-11-25         NaN
```

In [36]:
```
#16. Determine which sessions occurred on the same day each user registered

df6=pd.merge(users,sessions,how='outer',on='UserID')
df6[df6['Registered']==df6['SessionDate']]
```

Out[36]:
```
   UserID     User  Gender  Registered   Cancelled  SessionID SessionDate
```

```
2       2    Pedro    male  2010-08-01  2010-08-08        2.0  2010-08-01
8       4  Brielle  female  2013-07-17         NaN        9.0  2013-07-17
```

In [55]:
```
#17. Build a dataset with every possible (UserID, ProductID) pair (cross join)

df8=df5.iloc[:,2:4]
df8.dropna(inplace=True)
df8
```

Out[55]:

	UserID	ProductID
0	7.0	2.0
1	7.0	4.0
2	3.0	4.0
3	3.0	3.0
4	3.0	4.0
5	3.0	4.0
6	1.0	2.0
7	2.0	4.0
8	2.0	5.0

In [60]:
```
#18. Determine how much quantity of each product was purchased by each user

tolist=[(i,j) for i in df8['UserID'] for j in df8['ProductID']]
tolist=list(dict.fromkeys(tolist))
df9=pd.DataFrame(data=tolist,columns=['UserID','ProductID'])
df9.head(13)
```

Out[60]:

	UserID	ProductID
0	7.0	2.0
1	7.0	4.0
2	7.0	3.0
3	7.0	5.0
4	3.0	2.0
5	3.0	4.0
6	3.0	3.0
7	3.0	5.0
8	1.0	2.0
9	1.0	4.0
10	1.0	3.0
11	1.0	5.0
12	2.0	2.0

In [61]:
```
# Determine how much quantity of each product was purchased by each user

df1 = pd.DataFrame({'key': np.repeat(1, users.shape[0]), 'UserID': users.UserID})
df2 = pd.DataFrame({'key': np.repeat(1, products.shape[0]), 'ProductID': products.Proc
user_products = pd.merge(df1, df2,on='key')[['UserID', 'ProductID']]
pd.merge(user_products, transactions, how='left', on=['UserID', 'ProductID']).groupby
    Quantity=x.Quantity.sum()
))).reset_index().fillna(0)
```

Out[61]:

	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	6.0
14	3	5	0.0
15	4	1	0.0
16	4	2	0.0
17	4	3	0.0
18	4	4	0.0
19	4	5	0.0
20	5	1	0.0
21	5	2	0.0
22	5	3	0.0
23	5	4	0.0
24	5	5	0.0

In [62]:
```
# For each user, get each possible pair of pair transactions (TransactionID1, Transac

pd.merge(transactions, transactions, on='UserID')
```

Out[62]:

	TransactionID_x	TransactionDate_x	UserID	ProductID_x	Quantity_x	\
0	1	2010-08-21	7.0	2	1	
1	1	2010-08-21	7.0	2	1	
2	9	2015-04-24	7.0	4	3	
3	9	2015-04-24	7.0	4	3	
4	2	2011-05-26	3.0	4	1	
5	2	2011-05-26	3.0	4	1	
6	2	2011-05-26	3.0	4	1	
7	2	2011-05-26	3.0	4	1	
8	3	2011-06-16	3.0	3	1	
9	3	2011-06-16	3.0	3	1	
10	3	2011-06-16	3.0	3	1	
11	3	2011-06-16	3.0	3	1	
12	7	2013-12-30	3.0	4	1	
13	7	2013-12-30	3.0	4	1	
14	7	2013-12-30	3.0	4	1	
15	7	2013-12-30	3.0	4	1	

16	10	2016-05-08	3.0	4	4
17	10	2016-05-08	3.0	4	4
18	10	2016-05-08	3.0	4	4
19	10	2016-05-08	3.0	4	4
20	4	2012-08-26	1.0	2	3
21	5	2013-06-06	2.0	4	1
22	5	2013-06-06	2.0	4	1
23	6	2013-12-23	2.0	5	6
24	6	2013-12-23	2.0	5	6
25	8	2014-04-24	NaN	2	3

	TransactionID_y	TransactionDate_y	ProductID_y	Quantity_y
0	1	2010-08-21	2	1
1	9	2015-04-24	4	3
2	1	2010-08-21	2	1
3	9	2015-04-24	4	3
4	2	2011-05-26	4	1
5	3	2011-06-16	3	1
6	7	2013-12-30	4	1
7	10	2016-05-08	4	4
8	2	2011-05-26	4	1
9	3	2011-06-16	3	1
10	7	2013-12-30	4	1
11	10	2016-05-08	4	4
12	2	2011-05-26	4	1
13	3	2011-06-16	3	1
14	7	2013-12-30	4	1
15	10	2016-05-08	4	4
16	2	2011-05-26	4	1
17	3	2011-06-16	3	1
18	7	2013-12-30	4	1
19	10	2016-05-08	4	4
20	4	2012-08-26	2	3
21	5	2013-06-06	4	1
22	6	2013-12-23	5	6
23	5	2013-06-06	4	1
24	6	2013-12-23	5	6
25	8	2014-04-24	2	3

In [65]:
```
# Join each user to his/her first occuring transaction in the transactions table

data=pd.merge(users, transactions.groupby('UserID').first().reset_index(), how='left'
data
```

Out[65]:

	UserID	User	Gender	Registered	Cancelled	TransactionID	\
0	1	Charles	male	2012-12-21	NaN	4.0	
1	2	Pedro	male	2010-08-01	2010-08-08	5.0	
2	3	Caroline	female	2012-10-23	2016-06-07	2.0	

```
3       4   Brielle  female  2013-07-17         NaN            NaN
4       5  Benjamin    male  2010-11-25         NaN            NaN

  TransactionDate  ProductID  Quantity
0      2012-08-26        2.0       3.0
1      2013-06-06        4.0       1.0
2      2011-05-26        4.0       1.0
3             NaN        NaN       NaN
4             NaN        NaN       NaN
```

In [66]:
```
my_columns = list(data.columns)
my_columns
```

Out[66]:
```
['UserID',
 'User',
 'Gender',
 'Registered',
 'Cancelled',
 'TransactionID',
 'TransactionDate',
 'ProductID',
 'Quantity']
```

In [67]:
```
list(data.dropna(thresh=int(data.shape[0] * .9), axis=1).columns)
```

Out[67]:
```
['UserID', 'User', 'Gender', 'Registered']
```

In [68]:
```
missing_info= list(data.columns[data.isnull().any()])
missing_info
```

Out[68]:
```
['Cancelled', 'TransactionID', 'TransactionDate', 'ProductID', 'Quantity']
```

In [85]:
```
#//for col in missing_info:
num_missing = data[data[missing_info].isnull() == True].shape[0]
num_missing
```

Out[85]:
```
5
```