In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:

```
df = pd.read_csv("https://raw.githubusercontent.com/jackiekazil/data-wrangling/master/data/
df.head()
```

Out[2]:

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

In [4]:

```
df1 = pd.read_csv("https://raw.githubusercontent.com/kjam/data-wrangling-pycon/master/data/
df1.head()
```

Out[4]:

	STATION	STATION_NAME	DATE	PRCP	SNWD	SNOW	TMAX	TMIN	WDF(
0	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310101	46	-9999	-9999	-9999	-11	-999
1	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310102	107	-9999	-9999	50	11	-999
2	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310103	-9999	-9999	-9999	28	11	-999
3	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310105	13	-9999	-9999	39	11	-999
4	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310106	-9999	-9999	-9999	0	-22	-999

5 rows × 21 columns

Question1 - Get the metadata from the above files

In [6]:

```
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 [7]:

```
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
```

Question2 - Get the rows name from the above files

In [8]:

```
np.array(df.index)
```

Out[8]:

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

In [9]:

```
np.array(df1.index)
```

Out[9]:

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

Question3 - Change any column name from the above file

In [10]:

```
df.rename(columns={'Indicator':'Indicator_Id'}).head()
```

Out[10]:

	Indicator_Id	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Lo
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	Na
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	Na
2	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Andorra	Female	28	28.0	Na
3	Life expectancy at age 60 (years)	Published	2000	Europe	High-income	Andorra	Both sexes	23	23.0	Na
4	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	United Arab Emirates	Female	78	78.0	Na

Question4 - Change the column name from any of the above file and store the changes made permanently

In [11]:

```
df.rename(columns={'Indicator':'Indicator_Id'},inplace=True)
```

In [12]:

```
df.head()
```

Out[12]:

	Indicator_Id	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	Lo
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	Na
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0	Na
2	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Andorra	Female	28	28.0	Na
3	Life expectancy at age 60 (years)	Published	2000	Europe	High-income	Andorra	Both sexes	23	23.0	Na
4	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	United Arab Emirates	Female	78	78.0	Na

Question5 - Change the name of multiple columns

In [13]:

```
df.rename(columns={'PUBLISH STATES':'Publication Status','WHO region':'WHO Region'},inplace
```

In [14]:

```
df.head()
```

Out[14]:

	Indicator_Id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0
1	Life expectancy at birth (years)	Published	2000	Europe	High-income	Andorra	Both sexes	80	80.0
2	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Andorra	Female	28	28.0
3	Life expectancy at age 60 (years)	Published	2000	Europe	High-income	Andorra	Both sexes	23	23.0
4	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	United Arab Emirates	Female	78	78.0

Question6 - Arrange the values of a particular column in ascending order

In [15]:

```
df.sort_values('Year').head()
```

Out[15]:

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

Question7 - Arrange multiple column values in the ascending order

In [16]:

```
f.sort_values(by=['Indicator_Id','Country','Year','WHO Region','Publication Status'],ascendi
```

Out[16]:

	Indicator_Id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Num
2798	Healthy life expectancy (HALE) at birth (years)	Published	2000	Eastern Mediterranean	Low-income	Afghanistan	Male	45	
3363	Healthy life expectancy (HALE) at birth (years)	Published	2000	Eastern Mediterranean	Low-income	Afghanistan	Both sexes	45	
4456	Healthy life expectancy (HALE) at birth (years)	Published	2000	Eastern Mediterranean	Low-income	Afghanistan	Female	45	
2245	Healthy life expectancy (HALE) at birth (years)	Published	2012	Eastern Mediterranean	Low-income	Afghanistan	Both sexes	49	
3689	Healthy life expectancy (HALE) at birth (years)	Published	2012	Eastern Mediterranean	Low-income	Afghanistan	Female	49	

Question8 - Make Country as the first column of the Dataframe

In [17]:

```
col = list(df)
col.insert(0,col.pop(col.index('Country')))
col
df = df.loc[:,col]
df.head(5)
```

Out[17]:

	Country	Indicator_Id	Publication Status	Year	WHO Region	World Bank income group	Sex	Display Value	Numeric
0	Andorra	Life expectancy at birth (years)	Published	1990	Europe	High-income	Both sexes	77	77.0
1	Andorra	Life expectancy at birth (years)	Published	2000	Europe	High-income	Both sexes	80	80.0
2	Andorra	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Female	28	28.0
3	Andorra	Life expectancy at age 60 (years)	Published	2000	Europe	High-income	Both sexes	23	23.0
4	United Arab Emirates	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	Female	78	78.0

Question9 - Get the column array using variable

In [19]:

```
np.array(df['WHO Region'])
```

Out[19]:

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

Question10 - Get the subset rows 11,24,37

In [20]:

```
df.iloc[[11,24,37]]
```

Out[20]:

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

Question11 - Get the subset rows excluding 5,12,23 and 56

In [21]:

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

Out[21]:

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

Part- II

In [22]:

```
users = pd.read_csv("https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/use
users.head()
```

Out[22]:

	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 [23]:

```
sessions = pd.read_csv("https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/
sessions.head()
```

Out[23]:

	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 [24]:

```
products = pd.read_csv("https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/
products.head()
```

Out[24]:

	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 [25]:

```
transactions = pd.read_csv("https://raw.githubusercontent.com/ben519/DataWrangling/master/D
transactions.head()
```

Out[25]:

	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

Question12 - Join users to transactions, keeping all rows from transactions and only matching rows from user(left join)

In [26]:

```
users.columns
```

Out[26]:

```
Index(['UserID', 'User', 'Gender', 'Registered', 'Cancelled'], dtype='objec
t')
```

In [27]:

```
transactions.columns
```

Out[27]:

```
Index(['TransactionID', 'TransactionDate', 'UserID', 'ProductID', 'Quantit
y'], dtype='object')
```

In [28]:

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

Out[28]:

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

In [29]:

```
transactions = transactions.drop(columns=['User','Gender','Registered','Cancelled'])
transactions
```

Out[29]:

	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

Question13 - which transactions have a UserID not in users

In [30]:

```
transactions.isnull().sum()
```

Out[30]:

```
TransactionID      0
TransactionDate    0
UserID             1
ProductID          0
Quantity           0
dtype: int64
```

In [31]:

```
transactions[~transactions['UserID'].isin(users['UserID'])]
```

Out[31]:

	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

Question14 - Join Users to transactions, keeping only rows from transactions and users that match via UserID(inner join)

In [32]:

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

Out[32]:

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

Question15 - Join users to transactions, displayying all matching rows AND all non-matching rows(full outer join)

In [33]:

```
transactions.merge(users, how='outer', on='UserID')
```

Out[33]:

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

Question16 - Determine which sessions occured on the same day each user registered

In [35]:

```
pd.merge(left=users, right = sessions, how='inner', left_on=['UserID', 'Registered'], right
```

Out[35]:

	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

Question17 - Build a dataset with every possible(UserID, ProductID)pair(cross join)

In [36]:

```
userDF = users.loc[:,'UserID':'UserID']
```

In [37]:

```
productDF = products.loc[:,'ProductID':'ProductID']
```

In [38]:

```
index = pd.MultiIndex.from_product([userDF.UserID, productDF.ProductID], names = ["UserID",
userIDproductIDdf = pd.DataFrame(index = index).reset_index()
print(userIDproductIDdf)
```

```
    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
23       5          4
24       5          5
```

Question18 - Determine how much quantity of each product purchased by each user

In [39]:

```
user_product_Transdf = pd.merge(userIDproductIDdf, transactions, how='left', on=['UserID','
#print(user_product_Transdf) #Print the joined dataframe result

#Drop the unwanted column from the result dataframe and set to a new dataframe
user_product_TransdfResult = user_product_Transdf.drop(['TransactionID', 'TransactionDate']

#Set NaN values to zero(0) from result dataframe and set to a new dataframe
user_product_TransdfResult1 = user_product_TransdfResult.fillna(0)

print(user_product_TransdfResult1) # Print the fnal result
```

```
    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
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
```

Question19 -For each user, get each possible pair of pair transactions (TransactionID1, TransacationID2)

In [40]:

```
#transactions.head()
transactions1 = transactions.drop(['UserID'], axis=1)
pairOfTransdf = pd.merge(transactions, transactions, how='left', on=['UserID'])
print(pairOfTransdf.sort_values(by=['UserID'],ascending=[False]))
```

```
    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
21                9        2015-04-24     7.0            4           3
20                9        2015-04-24     7.0            4           3
24               10        2016-05-08     3.0            4           4
23               10        2016-05-08     3.0            4           4
22               10        2016-05-08     3.0            4           4
18                7        2013-12-30     3.0            4           1
17                7        2013-12-30     3.0            4           1
16                7        2013-12-30     3.0            4           1
15                7        2013-12-30     3.0            4           1
25               10        2016-05-08     3.0            4           4
9                 3        2011-06-16     3.0            3           1
8                 3        2011-06-16     3.0            3           1
7                 3        2011-06-16     3.0            3           1
6                 3        2011-06-16     3.0            3           1
5                 2        2011-05-26     3.0            4           1
4                 2        2011-05-26     3.0            4           1
3                 2        2011-05-26     3.0            4           1
2                 2        2011-05-26     3.0            4           1
11                5        2013-06-06     2.0            4           1
13                6        2013-12-23     2.0            5           6
14                6        2013-12-23     2.0            5           6
12                5        2013-06-06     2.0            4           1
10                4        2012-08-26     1.0            2           3
19                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
21                9        2015-04-24            4           3
20                1        2010-08-21            2           1
24                7        2013-12-30            4           1
23                3        2011-06-16            3           1
22                2        2011-05-26            4           1
18               10        2016-05-08            4           4
17                7        2013-12-30            4           1
16                3        2011-06-16            3           1
15                2        2011-05-26            4           1
25               10        2016-05-08            4           4
9                10        2016-05-08            4           4
8                 7        2013-12-30            4           1
7                 3        2011-06-16            3           1
6                 2        2011-05-26            4           1
5                10        2016-05-08            4           4
4                 7        2013-12-30            4           1
3                 3        2011-06-16            3           1
2                 2        2011-05-26            4           1
11                5        2013-06-06            4           1
13                5        2013-06-06            4           1
14                6        2013-12-23            5           6
12                6        2013-12-23            5           6
```

```
10              4        2012-08-26             2           3
19              8        2014-04-24             2           3
```

Question20 -Join each user to his/her first occuring transaction in the transactions table

In [41]:

```
#users.head()
#transactions.head(10)

# first select the first transaction for each user
firstOccur_transactions = transactions.groupby('UserID').first().reset_index()

#drop the unwanted column(s)
firstOccur_transactions = firstOccur_transactions.drop(['TransactionDate'], axis=1)

firstOccur_transactionsByEachUser = pd.merge(users, firstOccur_transactions, how='left', or
print(firstOccur_transactionsByEachUser)
```

```
   UserID      User  Gender  Registered   Cancelled  TransactionID  ProductI
D  \
0       1   Charles    male  2012-12-21         NaN            4.0        2.
0
1       2     Pedro    male  2010-08-01  2010-08-08            5.0        4.
0
2       3  Caroline  female  2012-10-23  2016-06-07            2.0        4.
0
3       4   Brielle  female  2013-07-17         NaN            NaN        Na
N
4       5  Benjamin    male  2010-11-25         NaN            NaN        Na
N

   Quantity
0       3.0
1       1.0
2       1.0
3       NaN
4       NaN
```

Question21 - Test to see if we can drop columns

In [42]:

```
data = firstOccur_transactionsByEachUser
my_columns = list(data.columns)
my_columns
['UserID',
'User',
'Gender',
'Registered',
'Cancelled',
'TransactionID',
'TransactionDate',
'ProductID',
'Quantity']
list(data.dropna(thresh=int(data.shape[0] * .9), axis=1).columns) #set threshold to drop NA
['UserID', 'User', 'Gender', 'Registered']
missing_info = list(data.columns[data.isnull().any()])
missing_info
['Cancelled', 'TransactionID', 'TransactionDate', 'ProductID', 'Quantity']
for col in missing_info:
    num_missing = data[data[col].isnull() == True].shape[0]
    print('number missing for column {}: {}'.format(col, num_missing))

print('--'*50)
print('Output: Count of missing data')
#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
for col in missing_info:
    num_missing = data[data[col].isnull() == True].shape[0]
    print('number missing for column {}: {}'.format(col, num_missing)) #count of missing da

print('--'*50)
for col in missing_info:
    percent_missing = data[data[col].isnull() == True].shape[0] / data.shape[0]
    print('percent missing for column {}:{}'.format(col, str(percent_missing)))

#Output of percentage missing data
#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
```

```
number missing for column Cancelled: 3
number missing for column TransactionID: 2
number missing for column ProductID: 2
number missing for column Quantity: 2
----------------------------------------------------------------------------------
------------------------
Output: Count of missing data
number missing for column Cancelled: 3
number missing for column TransactionID: 2
number missing for column ProductID: 2
number missing for column Quantity: 2
----------------------------------------------------------------------------------
------------------------
percent missing for column Cancelled:0.6
percent missing for column TransactionID:0.4
```

```
percent missing for column ProductID:0.4
percent missing for column Quantity:0.4
```

In [ ]: