Project - I - Answer

By Prakash Ghosh

In [1]:
```
# import the required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
# Create the first DataFrame from the given URL
df = pd.read_csv('https://raw.githubusercontent.com/jackiekazil/data-wrangling/master/data/chp3/data-text.csv')
df.head(2)
```

Out[2]:

	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 [3]:
```
# Create the Second DataFrame from the given URL
df1 =pd.read_csv('https://raw.githubusercontent.com/kjam/data-wrangling-pycon/master/data/berlin_weather_oldest.csv')
df1.head(2)
```

Out[3]:

	STATION	STATION_NAME	DATE	PRCP	SNWD	SNOW	TMAX	TMIN	WDFG	PGTM	...	WT09	WT07	WT01	WT0
0	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310101	46	-9999	-9999	-9999	-11	-9999	-9999	...	-9999	-9999	-9999	-9999
1	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310102	107	-9999	-9999	50	11	-9999	-9999	...	-9999	-9999	-9999	-9999

2 rows × 21 columns

1. Get the Metadata from the above files.

In [4]:
```
print('Metadata of the First Dataframe (created from data-text.csv)\n')
df.info()
print('\n\nMetadata of the Second Dataframe (created from berlin_weather_oldest.csv)\n')
df1.info()
```

```
Metadata of the First Dataframe (created from data-text.csv)

<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


Metadata of the Second Dataframe (created from berlin_weather_oldest.csv)

<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]:
```
print("Row names for data-text.csv:")
df.index.values
```

```
Row names for data-text.csv:
```

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

In [6]:
```
print("Row names for berlin_weather_oldest.csv:")
df1.index.values
```

```
Row names for berlin_weather_oldest.csv:
```

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

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

In [7]:
```
print("DataFrame existing column name (before rename):")
df.head(2)
```

```
DataFrame existing column name (before rename):
```

Out[7]:

	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 [8]:
```
print('Change the column Indicator to Indicator_id and dispaly (first 2 records)')
df.rename(columns = {'Indicator':'Indicator_id'}).head(2)
```

```
Change the column Indicator to Indicator_id and dispaly (first 2 records)
```

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

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

In [9]:
```
print('Change the column Indicator to Indicator_id permanently')
df.rename(columns = {'Indicator':'Indicator_id'},inplace=True)
df.head(2)
```

```
Change the column Indicator to Indicator_id permanently
```

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

5. Change the names of multiple columns.

In [10]:
```
print('Change the names of multiple columns permanently')
df.rename(columns = {'PUBLISH STATES':'Publication Status','WHO region':'WHO Region'},inplace=True)
df.head(2)
```

```
Change the names of multiple columns permanently
```

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]:
```
# Arrange Data set based on the value of Year in ascending order (Show first 5 Records)
df.sort_values(['Year'], ascending=True).head(5)
```

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

7. Arrange multiple column values in ascending order.

In [12]:
```
print("Arrange multiple column values in ascending order")
df.sort_values(by=['Indicator_id','Country','Year','WHO Region','Publication Status'], axis=0, inplace=True)
df.sort_index(inplace=True)
df[['Indicator_id','Country','Year','WHO Region','Publication Status']].head(3)
```

```
Arrange multiple column values in ascending order
```

Out[12]:

	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

8. Make country as the first column of the dataframe.

In [13]:
```
# Get the column names from the DataFrame into a list
lst_columns= list(df.columns)

# delete Country from the list and add Country in the first position
lst_columns.pop(lst_columns.index('Country'))
lst_columns.insert(0,'Country')

# Change the column name of the Datafreame
df[lst_columns].head(5)
```

Out[13]:

	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

9. Get the column array using a variable

In [14]:
```
print('Column array using a variable')
WHO_Region=df['WHO Region']
np.array(WHO_Region)
```

```
Column array using a variable
```

Out[14]:
```
array(['Europe', 'Europe', 'Europe', ..., 'Africa', 'Africa', 'Africa'],
      dtype=object)
```

10. Get the subset rows 11, 24, 37

In [15]:
```
print('Get subset data of rows no. 11, 24, 37')
# data based on index location
df.loc[[11,24,37]]
```

```
Get subset data of rows no. 11, 24, 37
```

Out[15]:

	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 [16]:
```
print('Get subset data of rows excluding rows no. 5, 12, 23, and 56')

lst_exclude = [5, 12, 23, 56]

# Use isin function with the list lst_exclude then show the Records except the list
df[~df.index.isin(lst_exclude)].head(15)
```

Get subset data of rows excluding rows no. 5, 12, 23, and 56

Out[16]:

	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
11	Life expectancy at birth (years)	Published	2012	Europe	High-income	Austria	Female	83	83.0	NaN	NaN	NaN

	Indicator_id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric	Low	High	Comments
13	Life expectancy at birth (years)	Published	2012	Europe	High-income	Belgium	Female	83	83.0	NaN	NaN	NaN
14	Life expectancy at birth (years)	Published	2000	Eastern Mediterranean	High-income	Bahrain	Male	73	73.0	NaN	NaN	NaN
15	Life expectancy at birth (years)	Published	1990	Eastern Mediterranean	High-income	Bahrain	Female	74	74.0	NaN	NaN	NaN
16	Life expectancy at age 60 (years)	Published	1990	Eastern Mediterranean	High-income	Bahrain	Male	17	17.0	NaN	NaN	NaN

Load datasets from CSV

In [17]:
```
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 [18]:
```
# users DataFrame
users.head(2)
```

Out[18]:

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

In [19]:
```
# sessions DataFrame
sessions.head(2)
```

Out[19]:

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

In [20]:
```
# products DataFrame
products.head(2)
```

Out[20]:

	ProductID	Product	Price
0	1	A	14.16
1	2	B	33.04

In [21]:
```
# transactions DataFrame
transactions.head(2)
```

Out[21]:

	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 [22]:
```
# Left Join users to transactions (UserID)
transactions.merge(users, how='left', on=['UserID'])
```

Out[22]:

	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 [23]:
```
# transactions where UserID not in users Dataframe
transactions[~transactions.UserID.isin(users.UserID)]
```

Out[23]:

	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 [24]:
```
# Inner Join users to transactions (UserID)
transactions.merge(users, how='inner', on=['UserID'])
```

Out[24]:

	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 [25]:
```
# Full Outer Join users to transactions (UserID)
transactions.merge(users, how='outer', on=['UserID'])
```

Out[25]:

	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 [26]:
```
# Inner Join users to sessions on (UserID=UserID) and (Registered=SessionDate)
users.merge(sessions, how='inner', left_on=['UserID','Registered'],right_on=['UserID','SessionDate'])
```

Out[26]:

	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

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

In [27]:
```
# Cross Join users to products
# Use a dummy column with same value in both the Dataframe and join based on that column
users['join_key']='-1'
products['join_key']='-1'
df_users_products = users.merge(products, on=['join_key'])

# drop the dummy columns
users=users.drop('join_key', axis=1)
products=products.drop('join_key', axis=1)

# Show the Result from Join
df_users_products[["UserID","ProductID"]]
```

Out[27]:

	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

18. Determine how much quantity of each product was purchased by each user

In [28]:
```
# Left Join with transactions group by 'UserID','ProductID' fill NaN Quantity by 0
df_users_products_Qty = df_users_products.merge(transactions, how='left', \
                        on=['UserID','ProductID']).groupby(['UserID', 'ProductID'] \
                        ).apply(lambda x: pd.Series(dict(Quantity=x.Quantity.sum())))

df_users_products_Qty.reset_index().fillna(0)
```

Out[28]:

	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

	UserID	ProductID	Quantity
23	5	4	0.0
24	5	5	0.0

19. For each user, get each possible pair of pair transactions (TransactionID1, TransacationID2)

In [29]:
```
# For each user, each possible pair of pair transactions
transactions.merge(transactions,on="UserID")
```

Out[29]:

	TransactionID_x	TransactionDate_x	UserID	ProductID_x	Quantity_x	TransactionID_y	TransactionDate_y	ProductID_y	Quantity
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
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

20. Join each user to his/her first occuring transaction in the transactions table

In [30]:
```
# Get first Transaction of each user, use it for left join with user

df_users_first_transaction=users.merge((transactions.groupby('UserID').first()), how ="left", on="UserID")
df_users_first_transaction
```

Out[30]:

	UserID	User	Gender	Registered	Cancelled	TransactionID	TransactionDate	ProductID	Quantity
0	1	Charles	male	2012-12-21	NaN	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	NaN	NaN	NaN	NaN	NaN
4	5	Benjamin	male	2010-11-25	NaN	NaN	NaN	NaN	NaN

21. Test to see if we can drop columns

In [31]:
```
# Get cloumn's list

my_columns= list(df_users_first_transaction)
my_columns
```

Out[31]:
```
['UserID',
 'User',
 'Gender',
 'Registered',
 'Cancelled',
 'TransactionID',
 'TransactionDate',
 'ProductID',
 'Quantity']
```

In [32]:
```
# set threshold to drop NAs

list(df_users_first_transaction.dropna(thresh=int(df_users_first_transaction.shape[0] * .9), axis=1).columns)
```

Out[32]:
```
['UserID', 'User', 'Gender', 'Registered']
```

In [33]:
```
missing_info = list(df_users_first_transaction.columns[df_users_first_transaction.isnull().any()])
missing_info
```

Out[33]:
```
['Cancelled', 'TransactionID', 'TransactionDate', 'ProductID', 'Quantity']
```

In [34]:
```
print("Output: Count of missing data\n")
for col in missing_info:
    num_missing = df_users_first_transaction[df_users_first_transaction[col].isnull() == True].shape[0]
    print('number missing for column {}: {}'.format(col, num_missing))
```

```
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
```

In [35]:
```
print("Output of percentage missing data\n")
for col in missing_info:
    percent_missing = df_users_first_transaction[df_users_first_transaction[col].isnull() \
                          == True].shape[0]/df_users_first_transaction.shape[0]
    print('percent missing for column {}: {}'.format(col, 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
```