Problem Statement

Read the 2 csv files and show head(2)

In [1]:

```
# Import Libraries - Modules
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Read & Display details from the 1st csv
df = pd.read_csv("https://raw.githubusercontent.com/jackiekazil/data-wrangling/ma
df.head(2)
```

Out[1]:

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

In [2]:

```
# Read & Display details from the 2nd  csv
df1 = pd.read_csv('https://raw.githubusercontent.com/kjam/data-wrangling-pycon/ma
df1.head(2)
```

Out[2]:

	STATION	STATION_NAME	DATE	PRCP	SNWD	SNOW	TMAX	TMIN	WDFG	I
0	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310101	46	-9999	-9999	-9999	-11	-9999	
1	GHCND:GME00111445	BERLIN TEMPELHOF GM	19310102	107	-9999	-9999	50	11	-9999	

2 rows × 21 columns

1. Get the Metadata from the above files.

In [3]:

```
#1. Get the Metadata from the 1st  files.
# Use - info() to display the meta data details for the first file
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]:

```
#1. Get the Metadata from the 2nd file.
# Use - info() to display the meta data details for the 2nd  file
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]:
```
# Row names of first file
df.index.values # getting the indexes / row name of the data
```

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

In [6]:
```
# Row names of Second file

df1.index.values  # getting the indexes / row name of the data
```

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

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

In [7]:
```
df.columns.values
```

Out[7]:
```
array(['Indicator', 'PUBLISH STATES', 'Year', 'WHO region',
       'World Bank income group', 'Country', 'Sex', 'Display Value',
       'Numeric', 'Low', 'High', 'Comments'], dtype=object)
```

In [8]:
```
df2 = df.rename(columns={'Indicator':'Indicator_Id'}) # renaming the columns by r
df2.head(2) # Changes are temporary
```

Out[8]:

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

In [9]:

```
df.head(2) # Changes are not reflected here so it's not permanent
```

Out[9]:

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

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

In [10]:

```
# Using inplace = True to make it permanent

df.rename(columns={'Indicator':'Indicator_Id'},inplace=True)
df.head(2) # Here changes are permanent
```

Out[10]:

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

5. Change the names of multiple columns.

In [11]:

```python
df.rename(columns={'PUBLISH STATES':'Publication Status','WHO region':'WHO Region
df.head(2) # Here changes are permanent
```

Out[11]:

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

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

In [12]:

```python
# Using sort_values() for Year to sort it in ascending order

df.sort_values(['Year'], ascending=[True])
```

3199	Life expectancy at age 60 (years)	Published	1990	Europe	Lower-middle-income	Republic of Moldova	Both sexes	17	17
1262	Life expectancy at age 60 (years)	Published	1990	Western Pacific	High-income	Cook Islands	Male	17	17
1259	Life expectancy at birth (years)	Published	1990	Western Pacific	High-income	Cook Islands	Male	67	67
3203	Life expectancy	Published	1990	South-East	Lower-middle	Maldives	Female	12	12

7. Arrange multiple column values in ascending order.

In [13]:
```
# Using sort_values() for Country , Year and Numeric to sort it in ascending orde

df.sort_values(['Country','Year','Numeric'], ascending=[True,True,False]) # False
```

	at birth (years)			Mediterranean	income		sexes		
2957	Life expectancy at birth (years)	Published	2000	Eastern Mediterranean	Low-income	Afghanistan	Male	54	54
2798	Healthy life expectancy (HALE) at birth (years)	Published	2000	Eastern Mediterranean	Low-income	Afghanistan	Male	45	45
3363	Healthy life expectancy (HALE) at birth (years)	Published	2000	Eastern Mediterranean	Low-income	Afghanistan	Both sexes	45	45
4456	Healthy life expectancy (HALE) at birth (years)	Published	2000	Eastern Mediterranean	Low-income	Afghanistan	Female	45	45

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

In [14]:
```
# df[['Country']]
# df =df[5] +df[:5]+df[6:]
# Putting the country column at first place
df[pd.unique(['Country'] + df.columns.values.tolist())] # if unique is not there
```

4	United Arab Emirates	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	Female	78	78
5	Antigua and Barbuda	Life expectancy at birth (years)	Published	2000	Americas	High-income	Male	72	72
6	Antigua and Barbuda	Life expectancy at age 60 (years)	Published	1990	Americas	High-income	Male	17	17
7	Antigua and Barbuda	Life expectancy at age 60 (years)	Published	2012	Americas	High-income	Both sexes	22	22
8	Australia	Life expectancy at birth (years)	Published	2012	Western Pacific	High-income	Male	81	81

9. Get the column array using a variable Expected Output:

In [15]:
```
# There could be multiple possible outcome for this, showing some of them
col1='Country'

df[[col1]].values # 2d array
```

Out[15]:
```
array([['Andorra'],
       ['Andorra'],
       ['Andorra'],
       ...,
       ['South Africa'],
       ['Zambia'],
       ['Zimbabwe']], dtype=object)
```

In [16]:
```
df[[col1]].values[:,0] # 1d array, all the rows and zerowth column
```

Out[16]:
```
array(['Andorra', 'Andorra', 'Andorra', ..., 'South Africa', 'Zambia',
       'Zimbabwe'], dtype=object)
```

In [17]:
```
df[col1].values # 1d array
```

Out[17]:
```
array(['Andorra', 'Andorra', 'Andorra', ..., 'South Africa', 'Zambia',
       'Zimbabwe'], dtype=object)
```

10. Get the subset rows 11, 24, 37 Expected Output:

In [18]:
```
#df.iloc
#df.loc both for accessing the rows
df.iloc[[11,23,37]] # here you are creating the list with [] sign
```

Out[18]:

	Indicator_Id	Publication Status	Year	WHO Region	World Bank income group	Country	Sex	Display Value	Numeric	Low
11	Life expectancy at birth (years)	Published	2012	Europe	High-income	Austria	Female	83	83.0	NaN
23	Life expectancy at age 60 (years)	Published	2000	Western Pacific	High-income	Brunei Darussalam	Female	22	22.0	NaN
37	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Cyprus	Female	26	26.0	NaN

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

```
In [19]: df.drop([5,12,23,56], axis=0) # axis=0 for rows, and axis=1 for column, by defaul
```

					group				
10	Life expectancy at birth (years)	Published	2012	Western Pacific	High-income	Australia	Both sexes	83	83
11	Life expectancy at birth (years)	Published	2012	Europe	High-income	Austria	Female	83	83
13	Life expectancy at birth (years)	Published	2012	Europe	High-income	Belgium	Female	83	83
	Life								

Load datasets from CSV - Users, Sessions, Products, Transactions

In [20]:

```
users = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/maste
sessions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/ma
products = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/ma
transactions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWranglin
print('-'*80)
print(users.head())
print('-'*80)
print(products.head())
print('-'*80)
print(sessions.head())
print('-'*80)
print(transactions.head())
```

```
-------------------------------------------------------------------------------
-
   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
-------------------------------------------------------------------------------
-
   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
-------------------------------------------------------------------------------
-
   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
-------------------------------------------------------------------------------
-
   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
```

In [21]:

```
users.info()
transactions.info()
sessions.info()
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
UserID        5 non-null int64
User          5 non-null object
Gender        5 non-null object
Registered    5 non-null object
Cancelled     2 non-null object
dtypes: int64(1), object(4)
memory usage: 280.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
TransactionID      10 non-null int64
TransactionDate    10 non-null object
UserID             9 non-null float64
ProductID          10 non-null int64
Quantity           10 non-null int64
dtypes: float64(1), int64(3), object(1)
memory usage: 480.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
SessionID      10 non-null int64
SessionDate    10 non-null object
UserID         10 non-null int64
dtypes: int64(2), object(1)
memory usage: 320.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
ProductID    5 non-null int64
Product      5 non-null object
Price        5 non-null float64
dtypes: float64(1), int64(1), object(1)
memory usage: 200.0+ bytes
```

In [22]:

```
# Data Preprocessing
# Changing the Object to date

users['Registered'] = pd.to_datetime(users.Registered)
sessions['SessionDate'] = pd.to_datetime(sessions.SessionDate)
transactions['TransactionDate'] = pd.to_datetime(transactions.TransactionDate)
```

In [23]:

```
users.info()
transactions.info()
sessions.info()
products.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
UserID        5 non-null int64
User          5 non-null object
Gender        5 non-null object
Registered    5 non-null datetime64[ns]
Cancelled     2 non-null object
dtypes: datetime64[ns](1), int64(1), object(3)
memory usage: 280.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 5 columns):
TransactionID      10 non-null int64
TransactionDate    10 non-null datetime64[ns]
UserID             9 non-null float64
ProductID          10 non-null int64
Quantity           10 non-null int64
dtypes: datetime64[ns](1), float64(1), int64(3)
memory usage: 480.0 bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 3 columns):
SessionID      10 non-null int64
SessionDate    10 non-null datetime64[ns]
UserID         10 non-null int64
dtypes: datetime64[ns](1), int64(2)
memory usage: 320.0 bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 3 columns):
ProductID    5 non-null int64
Product      5 non-null object
Price        5 non-null float64
dtypes: float64(1), int64(1), object(1)
memory usage: 200.0+ bytes
```

In [24]:

```
print(users.head())
print(transactions.head())
```

```
   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
   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
```

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

In [25]:

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

Out[25]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Canc
0	1	2010-08-21	7	2	1	NaN	NaN	NaT	
1	2	2011-05-26	3	4	1	Caroline	female	2012-10-23	201
2	3	2011-06-16	3	3	1	Caroline	female	2012-10-23	201
3	4	2012-08-26	1	2	3	Charles	male	2012-12-21	
4	5	2013-06-06	2	4	1	Pedro	male	2010-08-01	201
5	6	2013-12-23	2	5	6	Pedro	male	2010-08-01	201
6	7	2013-12-30	3	4	1	Caroline	female	2012-10-23	201
7	8	2014-04-24	NaN	2	3	NaN	NaN	NaT	
8	9	2015-04-24	7	4	3	NaN	NaN	NaT	
9	10	2016-05-08	3	4	4	Caroline	female	2012-10-23	201

13. Which transactions have a UserID not in users?

In [26]:
```
transactions['UserID'].isin(users['UserID']) #  Find which all rows doesn't in tr
```

Out[26]:
```
0    False
1     True
2     True
3     True
4     True
5     True
6     True
7    False
8    False
9     True
Name: UserID, dtype: bool
```

In [27]:
```
transactions[~transactions['UserID'].isin(users['UserID'])] # 7 AND NaN
```

Out[27]:

	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 [28]:
```
#transactions.merge(users, how='inner', on='UserID')
# transactions.merge(users, how='inner', left_on='UserID', right_on='UserID')
transactions.merge(users, how='inner', left_on =
                   'UserID', right_on = 'UserID')
```

Out[28]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Registered	Canc
0	2	2011-05-26	3	4	1	Caroline	female	2012-10-23	201
1	3	2011-06-16	3	3	1	Caroline	female	2012-10-23	201
2	7	2013-12-30	3	4	1	Caroline	female	2012-10-23	201
3	10	2016-05-08	3	4	4	Caroline	female	2012-10-23	201
4	4	2012-08-26	1	2	3	Charles	male	2012-12-21	
5	5	2013-06-06	2	4	1	Pedro	male	2010-08-01	201
6	6	2013-12-23	2	5	6	Pedro	male	2010-08-01	201

15. Join users to transactions, displaying all matching rows AND all

non-matching rows (full outer join)

In [29]:
```
pd.merge(left=transactions, right=users, how='outer', left_on = 'UserID', right_o
```

Out[29]:

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

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

In [30]:
```
pd.merge(left=users,right=sessions, how='inner',left_on=['UserID','Registered'],
```

Out[30]:

	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 [31]:
```
import numpy as np
df2=pd.DataFrame({'key': np.repeat(1,users.shape[0]), 'UserID':users.UserID}) # C
df2
```

Out[31]:

	UserID	key
0	1	1
1	2	1
2	3	1
3	4	1
4	5	1

In [32]:
```
df3=pd.DataFrame({'key':np.repeat(1,users.shape[0]), 'ProductID':products.Product
df3
```

Out[32]:

	ProductID	key
0	1	1
1	2	1
2	3	1
3	4	1
4	5	1

In [33]:
```
#products
```

In [34]:

```
# First Method

user_products = pd.merge(df2, df3,on='key')[['UserID', 'ProductID']]
user_products
```

Out[34]:

	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

In [35]:

```
### Second Method
users_1 = users
users_1['key'] = 0
products_1 = products
products_1['key'] = 0
pd.merge(users_1, products_1, on='key', how="outer")[['UserID', 'ProductID']]
```

Out[35]:

	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

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

In [36]:

```
# User Merge on the user_products and transactions with Left join combining/Group
# quantity of each product was purchased by each user
pd.merge(user_products, transactions, how='left', on=['UserID', 'ProductID']).gro
    Quantity=x.Quantity.sum()
))).reset_index().fillna(0)
```

Out[36]:

	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

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

In [37]:
```
#pd.merge(df2,df3, on ='key')[['UserID','ProductID']]
#df5= df4.merge(transactions, how='inner', left_on = 'UserID', right_on = 'UserID
#df5['UserID','ProductID_x',sum('Quantity')]
#df5
pd.merge(transactions, transactions, on='UserID')
```

Out[37]:

	TransactionID_x	TransactionDate_x	UserID	ProductID_x	Quantity_x	TransactionID_y	Transact
0	1	2010-08-21	7.0	2	1	1	2
1	1	2010-08-21	7.0	2	1	9	2
2	9	2015-04-24	7.0	4	3	1	2
3	9	2015-04-24	7.0	4	3	9	2
4	2	2011-05-26	3.0	4	1	2	2
5	2	2011-05-26	3.0	4	1	3	2
6	2	2011-05-26	3.0	4	1	7	2
7	2	2011-05-26	3.0	4	1	10	2
8	3	2011-06-16	3.0	3	1	2	2
9	3	2011-06-16	3.0	3	1	3	2
10	3	2011-06-16	3.0	3	1	7	2
11	3	2011-06-16	3.0	3	1	10	2
12	7	2013-12-30	3.0	4	1	2	2
13	7	2013-12-30	3.0	4	1	3	2
14	7	2013-12-30	3.0	4	1	7	2
15	7	2013-12-30	3.0	4	1	10	2
16	10	2016-05-08	3.0	4	4	2	2
17	10	2016-05-08	3.0	4	4	3	2
18	10	2016-05-08	3.0	4	4	7	2
19	10	2016-05-08	3.0	4	4	10	2
20	4	2012-08-26	1.0	2	3	4	2
21	5	2013-06-06	2.0	4	1	5	2
22	5	2013-06-06	2.0	4	1	6	2
23	6	2013-12-23	2.0	5	6	5	2
24	6	2013-12-23	2.0	5	6	6	2
25	8	2014-04-24	NaN	2	3	8	2

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

In [38]:
```
users.head(2)
```

Out[38]:

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

In [39]:
```
transactions.head(2)
```

Out[39]:

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

In [40]:
```
# First Method
#pd.merge(left=users,right=transactions, how='outer',left_on='UserID' , right_on=
df13 = users.merge(transactions, how='left',left_on='UserID' , right_on='UserID')
df13
```

Out[40]:

	UserID	User	Gender	Registered	Cancelled	key	TransactionID	TransactionDate	ProductII
0	1	Charles	male	2012-12-21	NaN	0	4.0	2012-08-26	2.
1	2	Pedro	male	2010-08-01	2010-08-08	0	5.0	2013-06-06	4.
3	3	Caroline	female	2012-10-23	2016-06-07	0	2.0	2011-05-26	4.
7	4	Brielle	female	2013-07-17	NaN	0	NaN	NaT	NaI
8	5	Benjamin	male	2010-11-25	NaN	0	NaN	NaT	NaI

In [41]:
```
users
```

Out[41]:

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

In [42]:
```
# Second Method
data=pd.merge(users, transactions.groupby('UserID').first().reset_index(), how='l
data
```

Out[42]:

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

21. Test to see if we can drop columns

In [43]:
```
data.drop('key', axis=1, inplace=True)
my_columns = list(data.columns)
my_columns
```

Out[43]:
```
['UserID',
 'User',
 'Gender',
 'Registered',
 'Cancelled',
 'TransactionID',
 'TransactionDate',
 'ProductID',
 'Quantity']
```

In [44]:
```
# Get the list of all columns without NAs & set threshold to drop NAs
list(data.dropna(thresh=int(data.shape[0] * .9), axis=1).columns)
```

Out[44]:
```
['UserID', 'User', 'Gender', 'Registered']
```

In [45]:
```
missing_info = list(data.columns[data.isnull().any()])
missing_info
```

Out[45]:
```
['Cancelled', 'TransactionID', 'TransactionDate', 'ProductID', 'Quantity']
```

In [46]:

```python
print("Count of missing data:\n")
for col in missing_info:
    num_missing = df13[df13[col].isnull() == True].shape[0]
    print('number missing for column {}: {}'.format(col, num_missing))
```

```
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 [47]:

```python
print("Percentage of missing data:\n")
for col in missing_info:
    percent_missing = df13[df13[col].isnull() == True].shape[0] / df13.shape[0]
    print('percent missing for column {}: {}'.format(col, percent_missing))
```

```
Percentage of 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
```