Project I Question Session 14: Project I

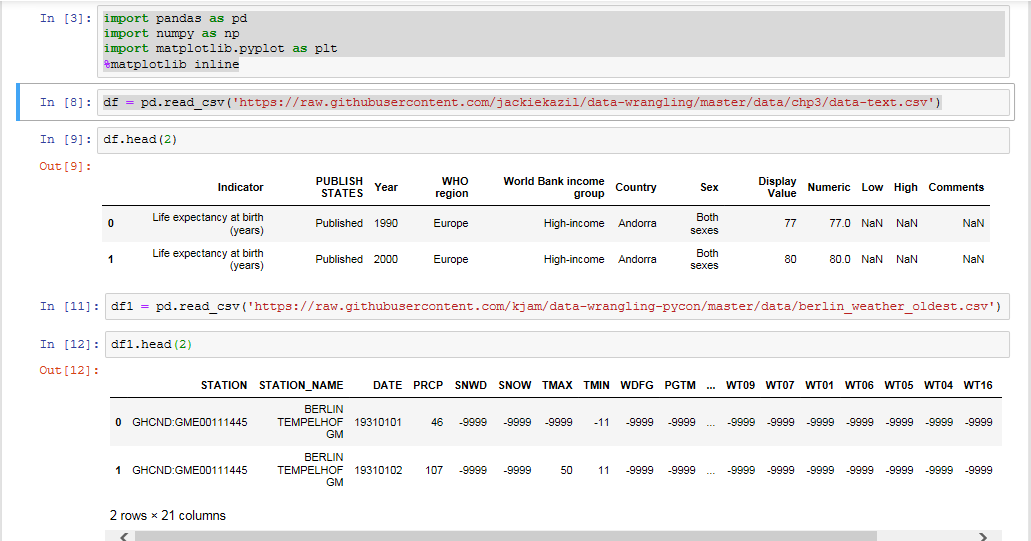
1. **Get the metadata from above files.**

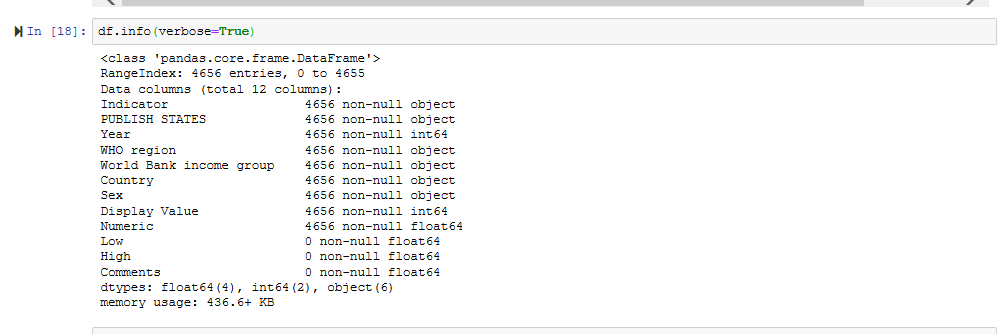
Code:

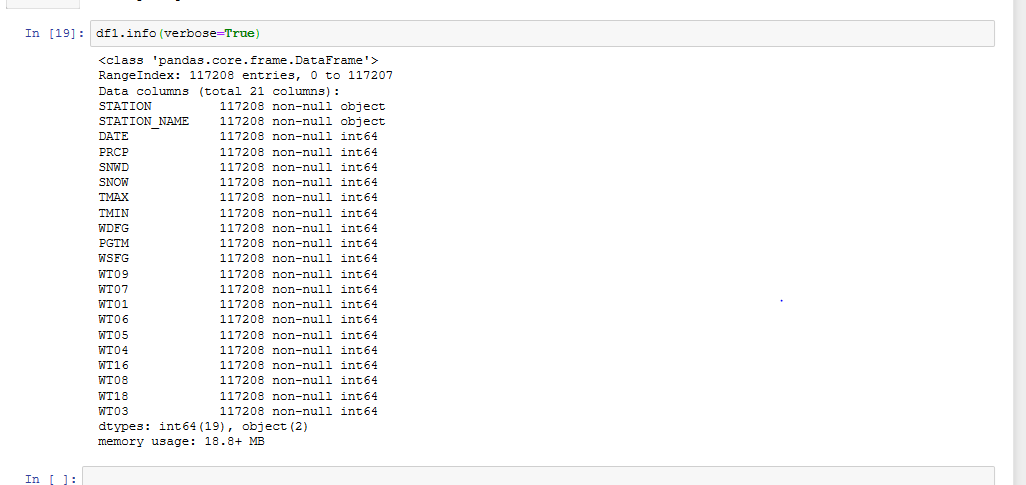
df.info(verbose=True)

df1.info(verbose=True)

Output:







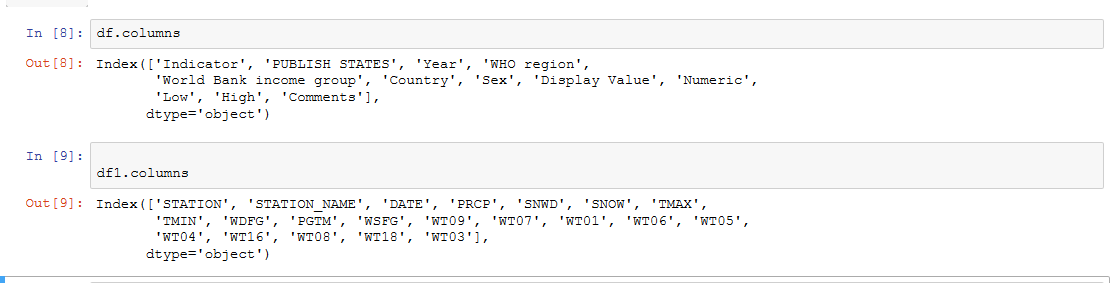
1. **Get the row names from the above files.**

**Code:**

df.columns

df1.columns

**Output:**

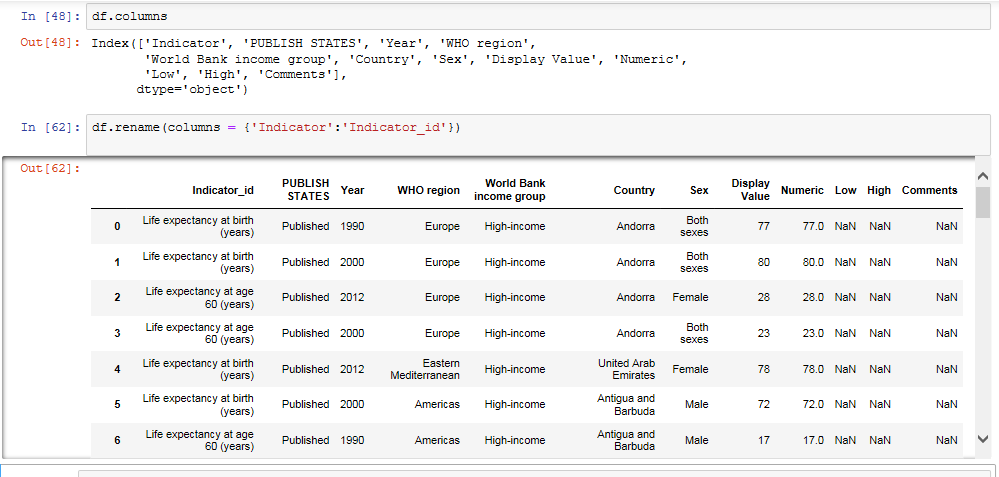


**3) Change the column name from any of the above file.**

**Code**:

df.rename(columns = {'Indicator':'Indicator\_id'})

**Output**:



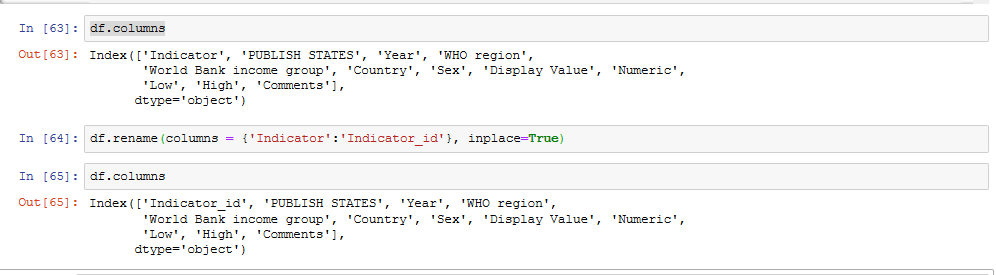
**4) Change the column name from any of the above file and store the changes made**

**Permanently.**

**Code**:

df.rename(columns = {'Indicator':'Indicator\_id'}, inplace=True)

**Output**:

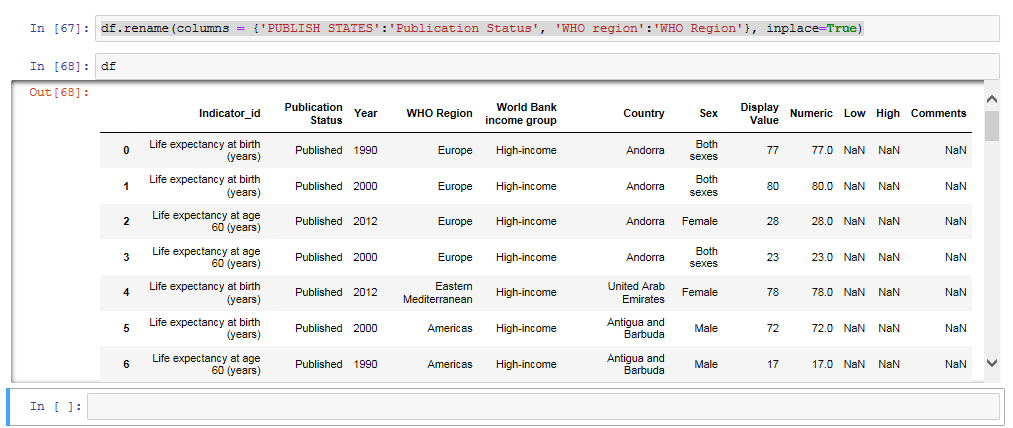


5**) Change the names of multiple Columns**

**Code**:

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

**Output**:

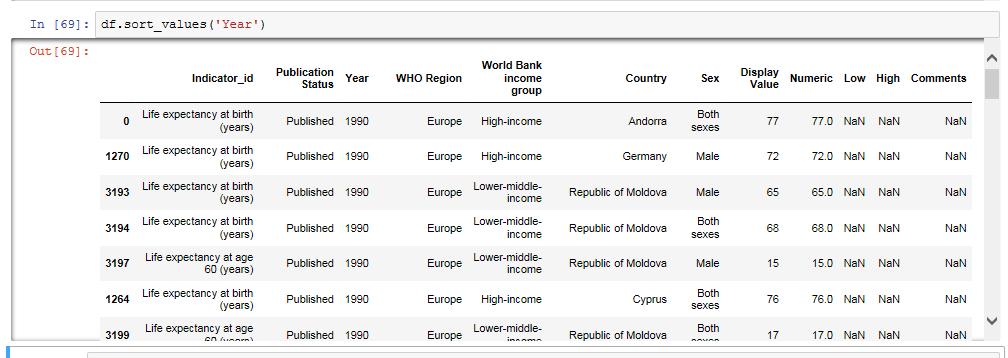


**6) Arrange values of a particular column in ascending order.**

**Code**:

df.sort\_values('Year')

**Output:**



**7) Arrange multiple column values in ascending order.**

**Code**:

df.sort\_values(by=['Indicator\_id','Country','WHO Region', 'Publication Status'],ascending=['True','True','True','True'])

**Output**:



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

**Code**:

df = df.reindex(['Country','Indicator\_id','Publication Status','Year','WHO Region','World Bank income group','Sex','Display Value','Numeric','Low','High', 'Comments '], axis=1)

**Output**:



**9) Get the column array using a variable.**

**Code**:

df["Country"].values

**Output**:

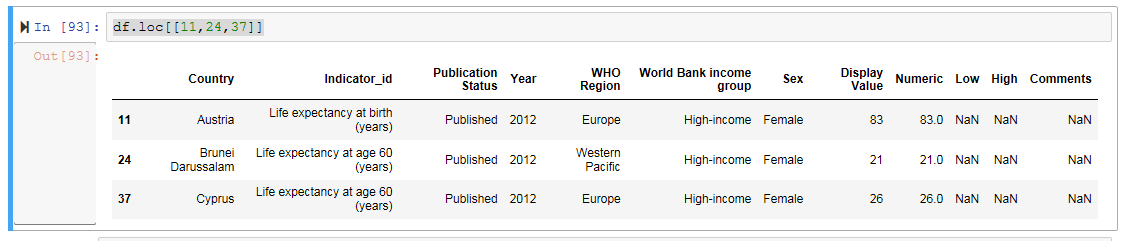


**10) Get the subset rows 11, 24, 37**

**Code**:

df.loc[[11,24,37]]

**Output:**



**11) Get the subset rows excluding 5, 12, 23, and 56**

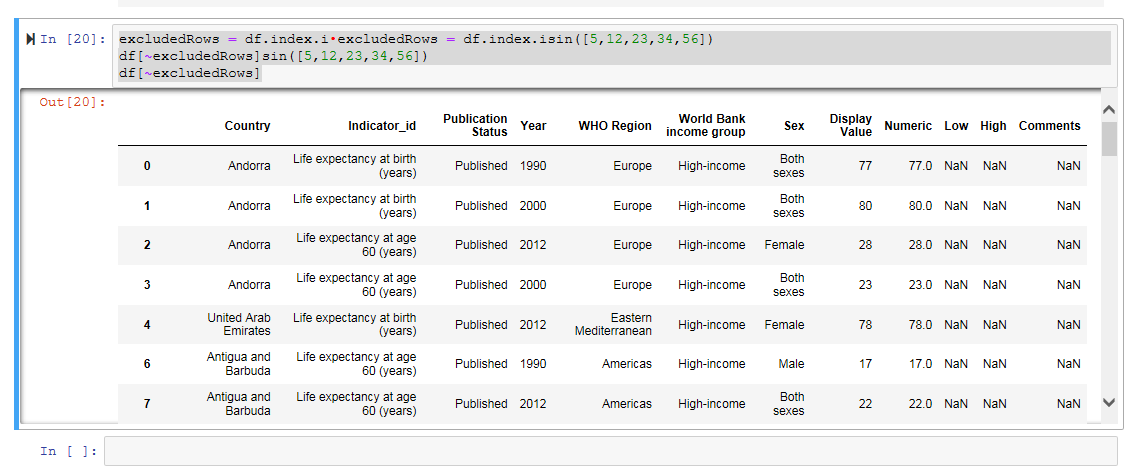
**Code:**

excludedRows = df.index.i​excludedRows = df.index.isin([5,12,23,34,56])

df[~excludedRows]sin([5,12,23,34,56])

df[~excludedRows]

**Output:**







Load datasets from CSV

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') users.head() sessions.head() transactions.head()

**12. Join users to transactions, keeping all rows from transactions and only matching rows**

**from users (left join)**

**Code:**

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')

users.head()

sessions.head()

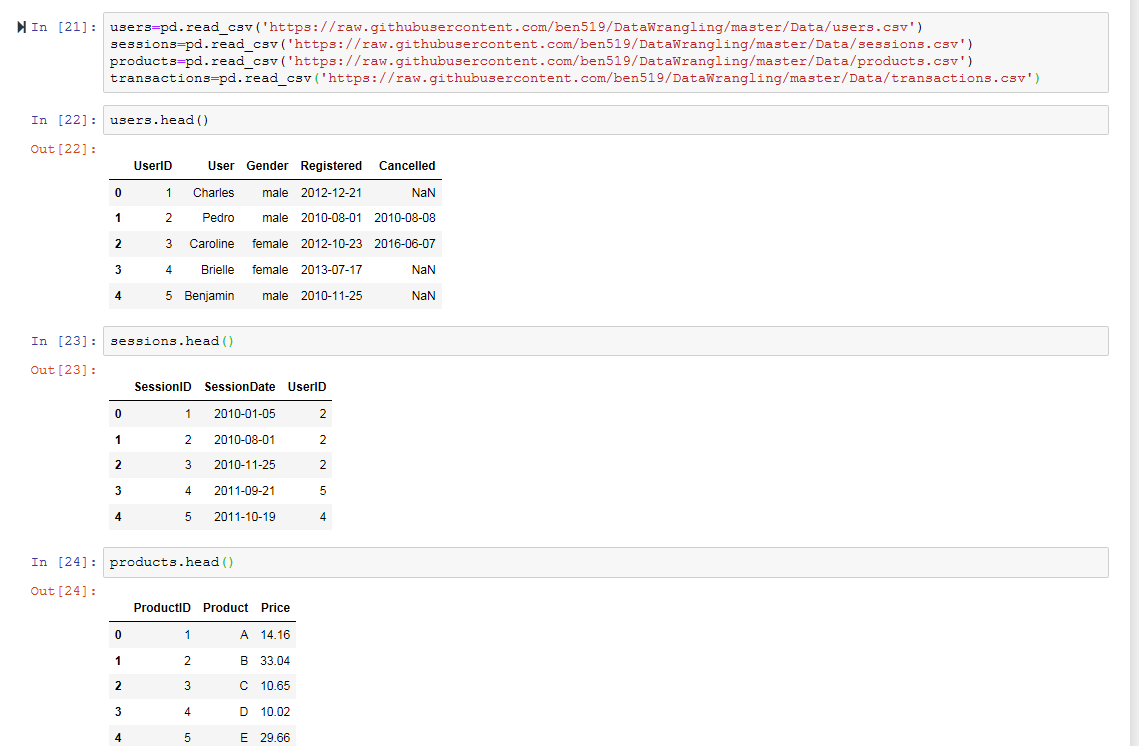
products.head()

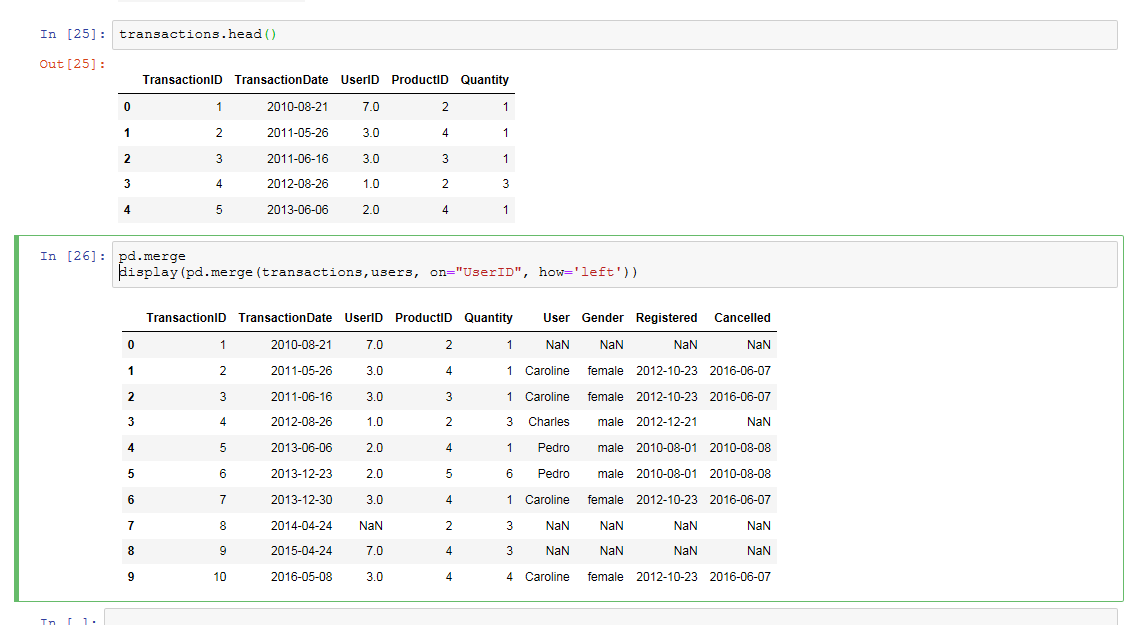
transactions.head()

pd.merge

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

**Output:**



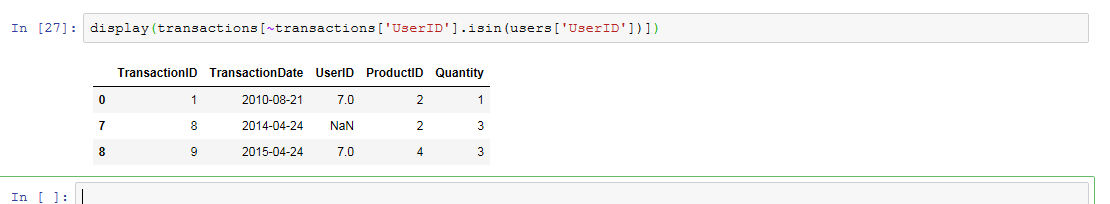


**13. Which transactions have a UserID not in users?**

**Code:**

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

**Output:**



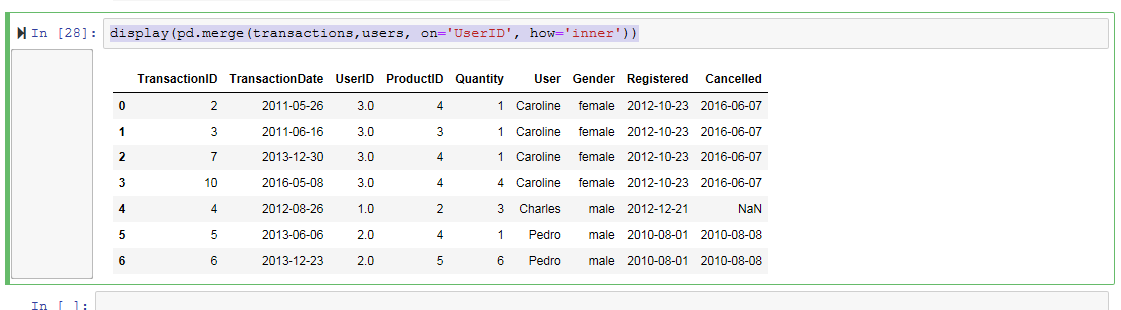
**14. Join users to transactions, keeping only rows from transactions and users that match**

**via UserID (inner join)**

**Code:**

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

**Output:**



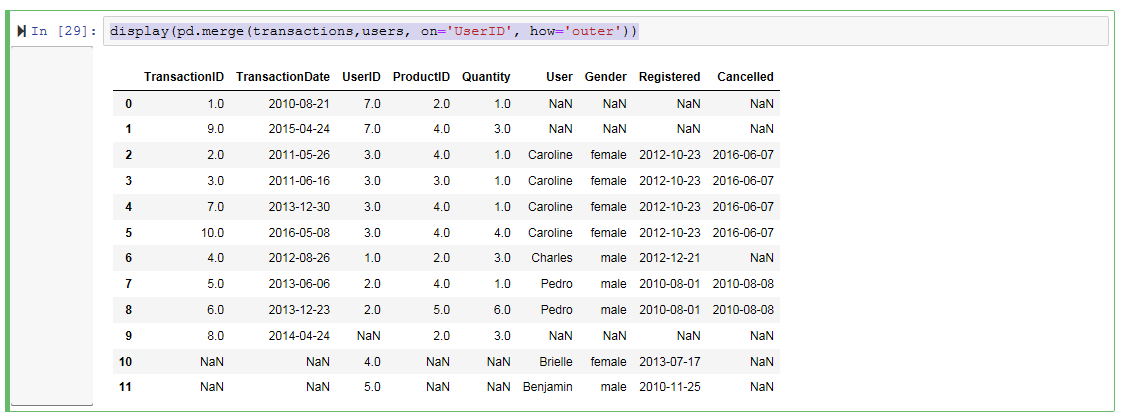
**15. Join users to transactions, displaying all matching rows AND all non-matching rows**

**(full outer join)**

**Code:**

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

**Output:**



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

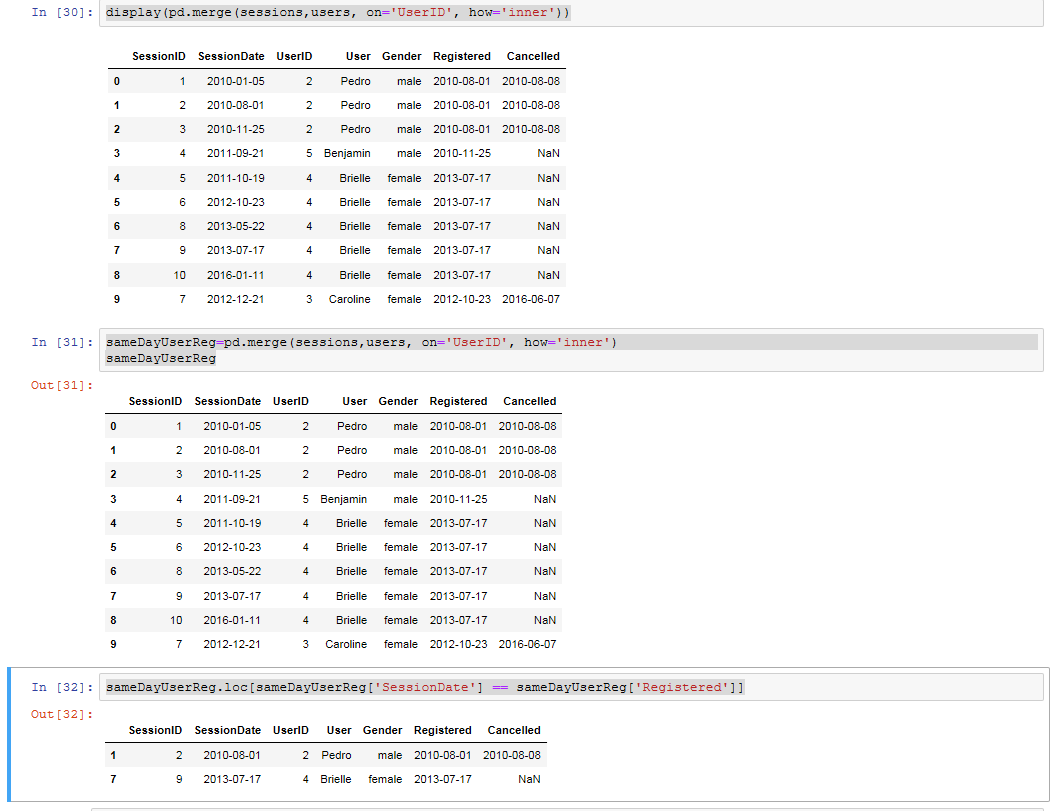
**Code:**

display(pd.merge(sessions,users, on='UserID', how='inner'))

sameDayUserReg=pd.merge(sessions,users, on='UserID', how='inner')

sameDayUserReg

sameDayUserReg.loc[sameDayUserReg['SessionDate'] == sameDayUserReg['Registered']]



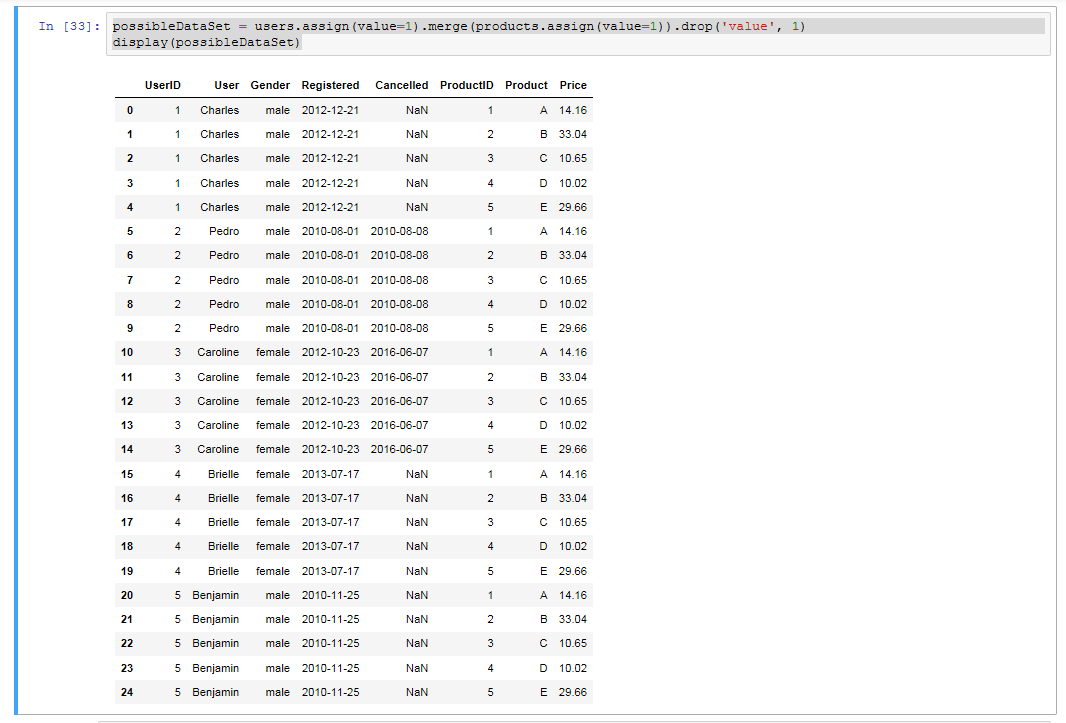
**17. Build a dataset with every possible (UserID, ProductID) pair (cross join)**

**Code:**

possibleDataSet = users.assign(value=1).merge(products.assign(value=1)).drop('value', 1)

display(possibleDataSet)

**Output:**

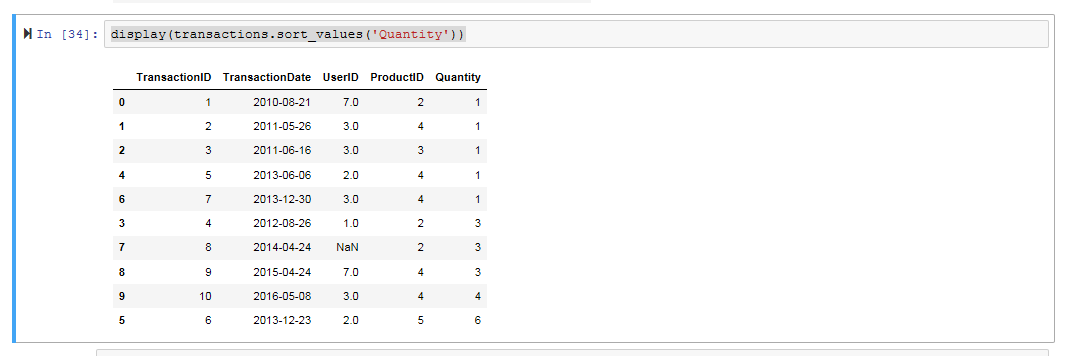


**Task 18. Determine how much quantity of each product was purchased by each user**

**Code:**

display(transactions.sort\_values('Quantity'))

**Output :**

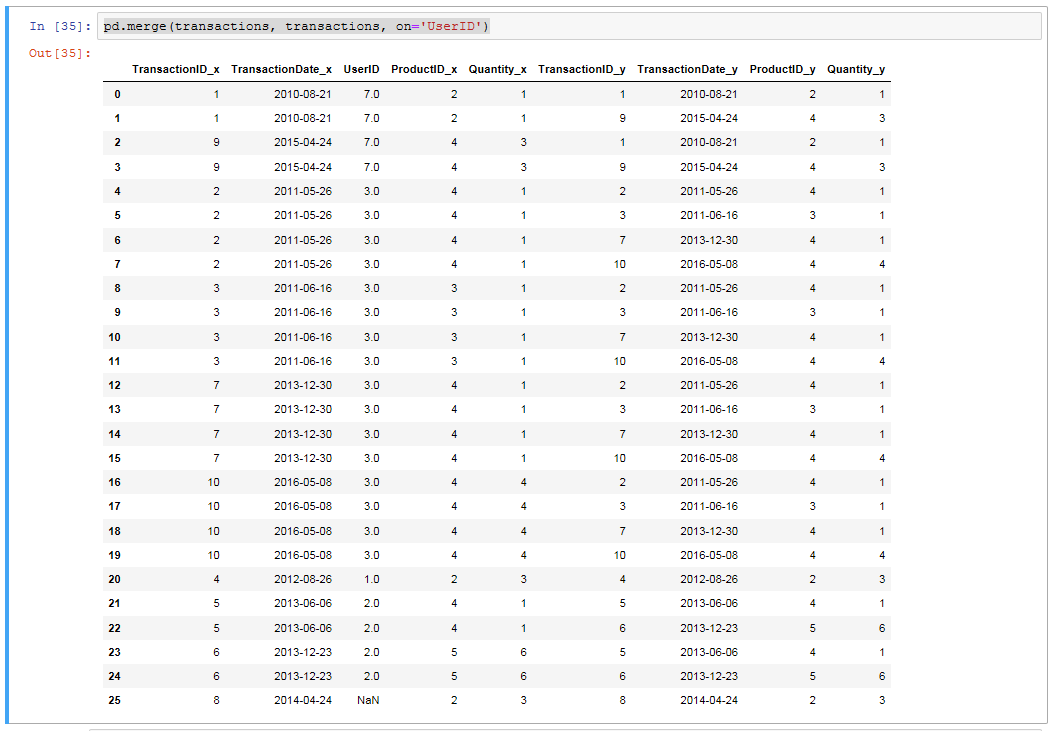


### Task 19. For each user, get each possible pair of pair transactions

**Code:**

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

**Output :**



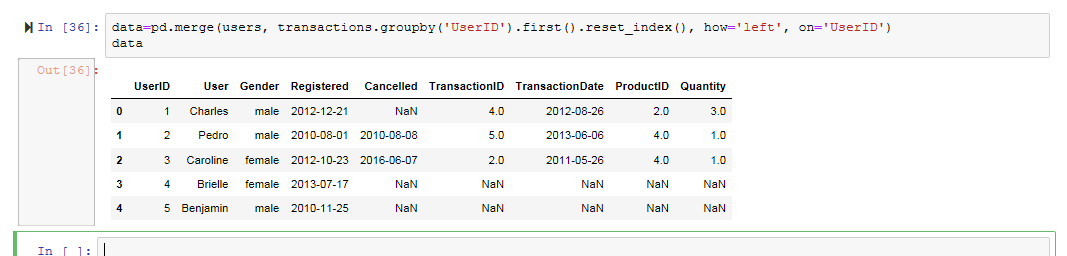
### Task20. Join each user to his/her first occuring transaction in the transactions table

**Code:**

data=pd.merge(users, transactions.groupby('UserID').first().reset\_index(), how='left', on='UserID')

data

**Output:**



### Task 21. Test to see if we can drop columns

**Code:**

columns = list(data.columns)

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

missingInfo = list(data.columns[data.isnull().any()])

missingInfo

for col in missingInfo:

missingNumber = data[data[col].isnull() == True].shape[0]

print('Missing Number for Col {}: {}'.format(col, missingNumber))

for col in missingInfo:

percentMissing = data[data[col].isnull() == True].shape[0] / data.shape[0]

print('Col Percent Missing {}: {}'.format(col, percentMissing))

**Output:**

