Please execute the code below and observe the output you get. Also, please learn how to use each of these statements to get a similar task done.

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

df = pd.read_csv('https://raw.githubusercontent.com/jackiekazil/data-wrangling/master/dat (https://raw.githubusercontent.com/jackiekazil/data-wrangling/master/dat) a/chp3/data-text.csv') df.head(2) df1 = pd.read_csv('https://raw.githubusercontent.com/kjam/data-wranglingpycon/master/d (https://raw.githubusercontent.com/kjam/data-wranglingpycon/master/d) ata/berlin_weather_oldest.csv') df1.head(2)

In [2]:

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

In [43]:

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

1. Get the Metadata from the above files.

In [44]:

```
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
```

2. Get the row names from the above files.

In [45]:

```
df.index.values
```

Out[45]:

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

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

In [46]:

```
df.rename(columns = {'Year':'years'})
```

```
...
```

In [47]:

```
df.head(1)
```

Out[47]:

	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

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

In [48]:

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

In [49]:

```
df.head(1)
```

Out[49]:

	Indicator_id	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

5. Change the names of multiple columns.

In [56]:

```
df.rename(columns = {'Country':'country','Sex':'sex'})
```

...

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

In [58]:

```
df.sort_values(by = 'Year')
```

Out[58]:

	Indicator_id	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeric	L
0	Life expectancy at birth (years)	Published	1990	Europe	High-income	Andorra	Both sexes	77	77.0	N
1270	Life expectancy at birth (years)	Published	1990	Europe	High-income	Germany	Male	72	72.0	N

7. Arrange multiple column values in ascending order.

In [59]:

```
df.sort_values(by = ['Year','Country'])
```

	at age 60 (years)			Mediterranean	income group					
299	Life expectancy at birth (years)	Published	1990	Europe	Lower-middle-income	Albania	Male	67	67.0	Na
689	Life expectancy at birth (years)	Published	1990	Europe	Lower-middle-income	Albania	Both sexes	69	69.0	Na
1522	Life expectancy at age 60 (years)	Published	1990	Europe	Lower-middle-income	Albania	Both sexes	16	16.0	Na

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

In [64]:

```
df.set_index('Country').reset_index()
```

	Andorra	at age 60 (years)	Published	2000	Europe	income	sexes			
4	United Arab Emirates	Life expectancy at birth (years)	Published	2012	Eastern Mediterranean	High-income	Female	78	78.0	Na
5	Antigua and Barbuda	Life expectancy at birth (years)	Published	2000	Americas	High-income	Male	72	72.0	Na

10. Get the subset rows 11, 24, 37

In [86]:

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

Out[86]:

	Indicator_id	PUBLISH STATES	Year	WHO region	World Bank income group	Country	Sex	Display Value	Numeri
11	Life expectancy at birth (years)	Published	2012	Europe	High-income	Austria	Female	83	83.0
24	Life expectancy at age 60 (years)	Published	2012	Western Pacific	High-income	Brunei Darussalam	Female	21	21.0
37	Life expectancy at age 60 (years)	Published	2012	Europe	High-income	Cyprus	Female	26	26.0

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

In [95]:

```
d1=df.index.isin([5,12,23,56])
df[~d1]
```

	at age 60 (years)				income					
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
6	Life expectancy at age 60 (years)	Published	1990	Americas	High-income	Antigua and Barbuda	Male	17	17.0	Na

Load datasets from CSV

users= pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/users.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 (https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/sessions.csv)')

products = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/products.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 (https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/transactions.csv)')

In [3]:

```
users= pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/user
sessions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/
products = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/Data/
transactions = pd.read_csv('https://raw.githubusercontent.com/ben519/DataWrangling/master/D
```

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

In [17]:

```
transactions.join(users,on = 'UserID',how='left',lsuffix='_transactions',rsuffix='_users')
```

Out[17]:

	TransactionID	TransactionDate	UserID_transactions	ProductID	Quantity	UserID_us
0	1	2010-08-21	7	2	1	NaN
1	2	2011-05-26	3	4	1	4
2	3	2011-06-16	3	3	1	4
3	4	2012-08-26	1	2	3	2
4	5	2013-06-06	2	4	1	3
5	6	2013-12-23	2	5	6	3
6	7	2013-12-30	3	4	1	4
7	8	2014-04-24	NaN	2	3	NaN
8	9	2015-04-24	7	4	3	NaN
9	10	2016-05-08	3	4	4	4

In [20]:

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

Out[20]:

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

In [45]:

```
import sqlite3 as db
```

In [46]:

```
conn = db.connect('test.db')
```

In [121]:

```
transactions.to_sql('trxn',conn,if_exists="replace")
users.to_sql('users',conn,if_exists="replace")
products.to_sql('products',conn,if_exists="replace")
sessions.to_sql('sessions',conn,if_exists="replace")
```

13. Which transactions have a UserID not in users?

In [53]:

```
pd.read_sql_query('select * from trxn where userID not in (select userID from users) or use
```

Out[53]:

	index	TransactionID	TransactionDate	UserID	ProductID	Quantity
0	0	1	2010-08-21	7.0	2	1
1	7	8	2014-04-24	NaN	2	3
2	8	9	2015-04-24	7.0	4	3

In [131]:

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

Out[131]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	ranked
0	1	2010-08-21	7.0	2	1	1.0
7	8	2014-04-24	NaN	2	3	NaN
8	9	2015-04-24	7.0	4	3	2.0

14. Join users to transactions, keeping only rows from transactions and users that match via UserID (inner join)

In [56]:

```
pd.read_sql_query('''
select t.* from trxn as t inner join users as u
on u.userID=t.userID''', conn)
```

Out[56]:

	index	TransactionID	TransactionDate	UserID	ProductID	Quantity
0	1	2	2011-05-26	3.0	4	1
1	2	3	2011-06-16	3.0	3	1
2	3	4	2012-08-26	1.0	2	3
3	4	5	2013-06-06	2.0	4	1
4	5	6	2013-12-23	2.0	5	6
5	6	7	2013-12-30	3.0	4	1
6	9	10	2016-05-08	3.0	4	4

In [132]:

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

Out[132]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	ranked	User	Gende
0	2	2011-05-26	3	4	1	1.0	Caroline	female
1	3	2011-06-16	3	3	1	2.0	Caroline	female
2	7	2013-12-30	3	4	1	3.0	Caroline	female
3	10	2016-05-08	3	4	4	4.0	Caroline	female
4	4	2012-08-26	1	2	3	1.0	Charles	male
5	5	2013-06-06	2	4	1	1.0	Pedro	male
6	6	2013-12-23	2	5	6	2.0	Pedro	male

15.Join users to transactions, displaying all matching rows AND all non-matching rows (full outer join)

In [60]:

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

Out[60]:

	TransactionID	TransactionDate	UserID	ProductID	Quantity	User	Gender	Reg
0	1.0	2010-08-21	7.0	2.0	1.0	NaN	NaN	Na
1	9.0	2015-04-24	7.0	4.0	3.0	NaN	NaN	Na
2	2.0	2011-05-26	3.0	4.0	1.0	Caroline	female	201
3	3.0	2011-06-16	3.0	3.0	1.0	Caroline	female	201
4	7.0	2013-12-30	3.0	4.0	1.0	Caroline	female	201
5	10.0	2016-05-08	3.0	4.0	4.0	Caroline	female	201
6	4.0	2012-08-26	1.0	2.0	3.0	Charles	male	201
7	5.0	2013-06-06	2.0	4.0	1.0	Pedro	male	201
8	6.0	2013-12-23	2.0	5.0	6.0	Pedro	male	201
9	8.0	2014-04-24	NaN	2.0	3.0	NaN	NaN	Na
10	NaN	NaN	4.0	NaN	NaN	Brielle	female	201
11	NaN	NaN	5.0	NaN	NaN	Benjamin	male	201

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

In [65]:

```
pd.read_sql_query('''select * from users as u inner join sessions as s
on u.userID=s.userID
and SessionDate = Registered
''', conn)
```

Out[65]:

	index	UserID	User	Gender	Registered	Cancelled	index	SessionID	SessionDate
0	1	2	Pedro	male	2010-08-01	2010-08-08	1	2	2010-08-01
1	3	4	Brielle	female	2013-07-17	None	8	9	2013-07-17

```
In [133]:
```

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

```
Out[133]:
```

	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 [68]:

```
pd.read_sql_query('''select u.userID,p.ProductID from users as u join products as p
''', conn)
```

Out[68]:

	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 [134]:

```
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.ProductID
pd.merge(df1, df2,on='key')[['UserID', 'ProductID']]
```

Out[134]:

	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 [73]:

```
pd.read_sql_query('''
select userId,ProductID,sum(Quantity)
from trxn as t
group by userID,ProductID
having t.userID is not null
''', conn)
```

Out[73]:

	UserID	ProductID	sum(Quantity)
0	1.0	2	3
1	2.0	4	1
2	2.0	5	6
3	3.0	3	1
4	3.0	4	6
5	7.0	2	1
6	7.0	4	3

```
In [136]:
```

```
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.ProductI
user_products = pd.merge(df1, df2,on='key')[['UserID', 'ProductID']]
pd.merge(user_products, transactions, how='left', on=['UserID', 'ProductID']).groupby(['Use
    Quantity=x.Quantity.sum()
))).reset_index()
```

```
Out[136]:
```

	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)

Question is not clear

In [130]:

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

Out[130]:

	TransactionID_x	TransactionDate_x	UserID	ProductID_x	Quantity_x	ranked_x	Tr
0	1	2010-08-21	7.0	2	1	1.0	1
1	1	2010-08-21	7.0	2	1	1.0	9
2	9	2015-04-24	7.0	4	3	2.0	1
3	9	2015-04-24	7.0	4	3	2.0	9
4	2	2011-05-26	3.0	4	1	1.0	2
5	2	2011-05-26	3.0	4	1	1.0	3
6	2	2011-05-26	3.0	4	1	1.0	7
7	2	2011-05-26	3.0	4	1	1.0	10
8	3	2011-06-16	3.0	3	1	2.0	2
9	3	2011-06-16	3.0	3	1	2.0	3
10	3	2011-06-16	3.0	3	1	2.0	7
11	3	2011-06-16	3.0	3	1	2.0	10
12	7	2013-12-30	3.0	4	1	3.0	2
13	7	2013-12-30	3.0	4	1	3.0	3
14	7	2013-12-30	3.0	4	1	3.0	7
15	7	2013-12-30	3.0	4	1	3.0	10
16	10	2016-05-08	3.0	4	4	4.0	2
17	10	2016-05-08	3.0	4	4	4.0	3
18	10	2016-05-08	3.0	4	4	4.0	7
19	10	2016-05-08	3.0	4	4	4.0	10
20	4	2012-08-26	1.0	2	3	1.0	4
21	5	2013-06-06	2.0	4	1	1.0	5
22	5	2013-06-06	2.0	4	1	1.0	6
23	6	2013-12-23	2.0	5	6	2.0	5
24	6	2013-12-23	2.0	5	6	2.0	6
25	8	2014-04-24	NaN	2	3	NaN	8

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

In [109]:

```
transactions['ranked']=transactions.groupby('UserID')['TransactionDate'].rank(ascending=1,m
```

In [110]:

```
df=transactions[transactions['ranked'] == 1]
```

In [114]:

```
users.merge(df,'left','UserID')
```

Out[114]:

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

21. Test to see if we can drop columns

In [119]:

```
users.drop('Gender',axis=1)
```

Out[119]:

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

In [ ]:

```
users.drop('Gender',axis=1,inplace=True)
```