iloc(), loc() for row

[] for column

# Merging Dataframes

import pandas as pd

df = pd.DataFrame([{'Name': 'Chris', 'Item Purchased': 'Sponge', 'Cost': 22.50},

{'Name': 'Kevyn', 'Item Purchased': 'Kitty Litter', 'Cost': 2.50},

{'Name': 'Filip', 'Item Purchased': 'Spoon', 'Cost': 5.00}],

index=['Store 1', 'Store 1', 'Store 2'])

df



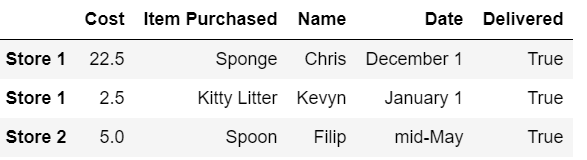
df['Date'] = ['December 1', 'January 1', 'mid-May'] # add a column

df



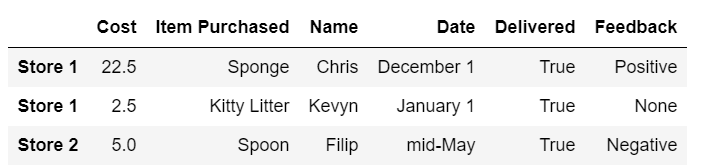
df['Delivered'] = True # add a column with default data

df



df['Feedback'] = ['Positive', None, 'Negative']

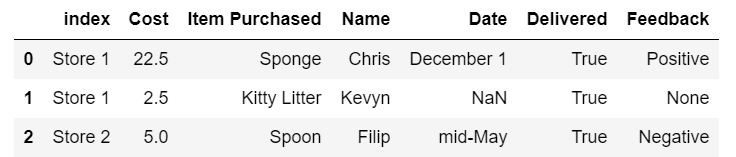
df

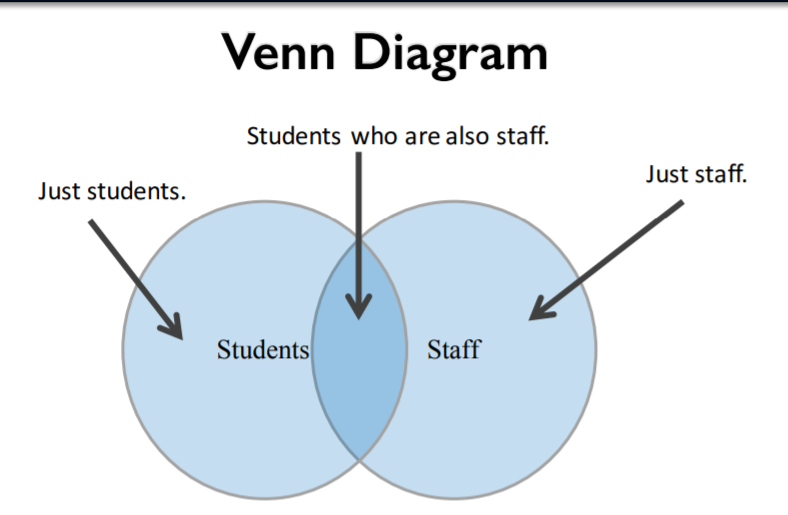


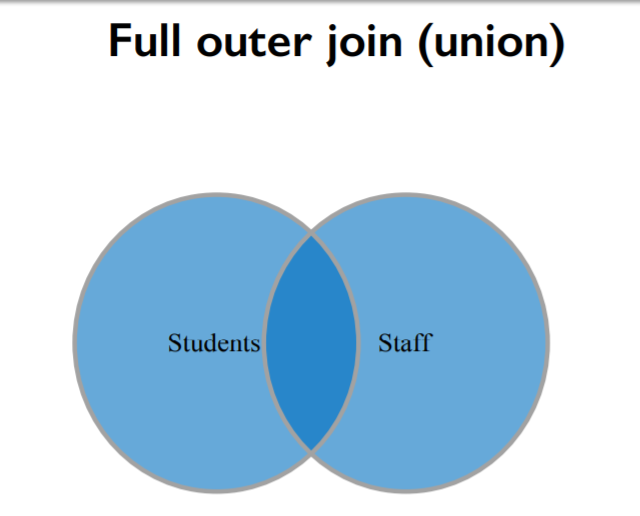
adf = df.**reset\_index**()

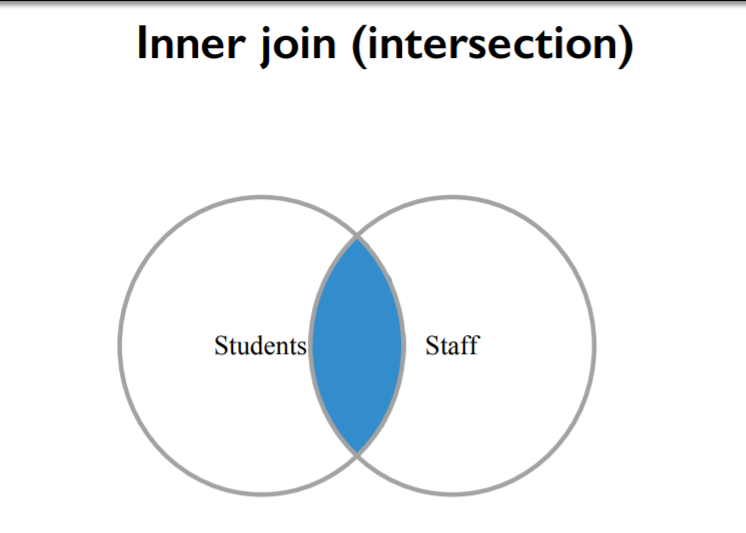
adf['Date'] = pd.Series({0: 'December 1', 2: 'mid-May'}) # assign by index

adf









(OR & AND)

staff\_df = pd.DataFrame([{'Name': 'Kelly', 'Role': 'Director of HR'},

{'Name': 'Sally', 'Role': 'Course liasion'},

{'Name': 'James', 'Role': 'Grader'}])

staff\_df = staff\_df.set\_index('Name')

student\_df = pd.DataFrame([{'Name': 'James', 'School': 'Business'},

{'Name': 'Mike', 'School': 'Law'},

{'Name': 'Sally', 'School': 'Engineering'}])

student\_df = student\_df.set\_index('Name')

print(staff\_df.head())

print()

print(student\_df.head())

Role

Name

Kelly Director of HR

Sally Course liasion

James Grader

School

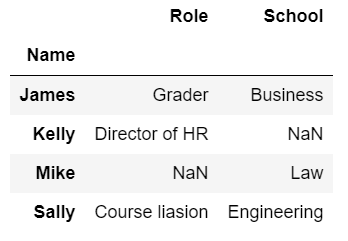
Name

James Business

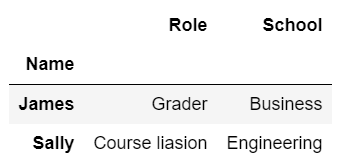
Mike Law

Sally Engineering

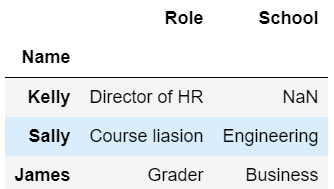
pd.**merge**(staff\_df, student\_df, how=**'outer'**, left\_index=True, right\_index=True) #outer join



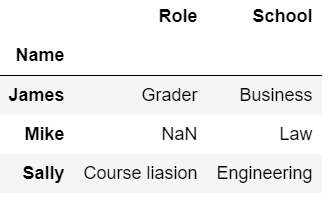
pd.**merge**(staff\_df, student\_df, how=**'inner'**, left\_index=True, right\_index=True) #inner join



pd.**merge**(staff\_df, student\_df, how=**'left'**, left\_index=True, right\_index=True) #left join



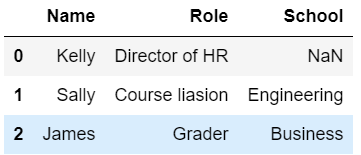
pd.**merge**(staff\_df, student\_df, how=**'right'**, left\_index=True, right\_index=True) #right join



staff\_df = staff\_df.reset\_index()

student\_df = student\_df.reset\_index()

pd.merge(staff\_df, student\_df, how='left', left\_on='Name', right\_on='Name') #left join on name, without default indexes



staff\_df = pd.DataFrame([{'Name': 'Kelly', 'Role': 'Director of HR', 'Location': 'State Street'},

{'Name': 'Sally', 'Role': 'Course liasion', 'Location': 'Washington Avenue'},

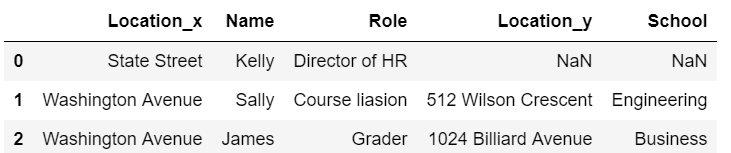
{'Name': 'James', 'Role': 'Grader', 'Location': 'Washington Avenue'}])

student\_df = pd.DataFrame([{'Name': 'James', 'School': 'Business', 'Location': '1024 Billiard Avenue'},

{'Name': 'Mike', 'School': 'Law', 'Location': 'Fraternity House #22'},

{'Name': 'Sally', 'School': 'Engineering', 'Location': '512 Wilson Crescent'}])

pd.merge(staff\_df, student\_df, how='left', left\_on='Name', right\_on='Name') # same index if over 2 values for 1



staff\_df = pd.DataFrame([{'First Name': 'Kelly', 'Last Name': 'Desjardins', 'Role': 'Director of HR'},

{'First Name': 'Sally', 'Last Name': 'Brooks', 'Role': 'Course liasion'},

{'First Name': 'James', 'Last Name': 'Wilde', 'Role': 'Grader'}])

student\_df = pd.DataFrame([{'First Name': 'James', 'Last Name': 'Hammond', 'School': 'Business'},

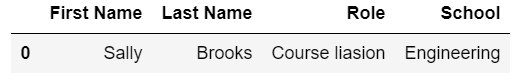
{'First Name': 'Mike', 'Last Name': 'Smith', 'School': 'Law'},

{'First Name': 'Sally', 'Last Name': 'Brooks', 'School': 'Engineering'}])

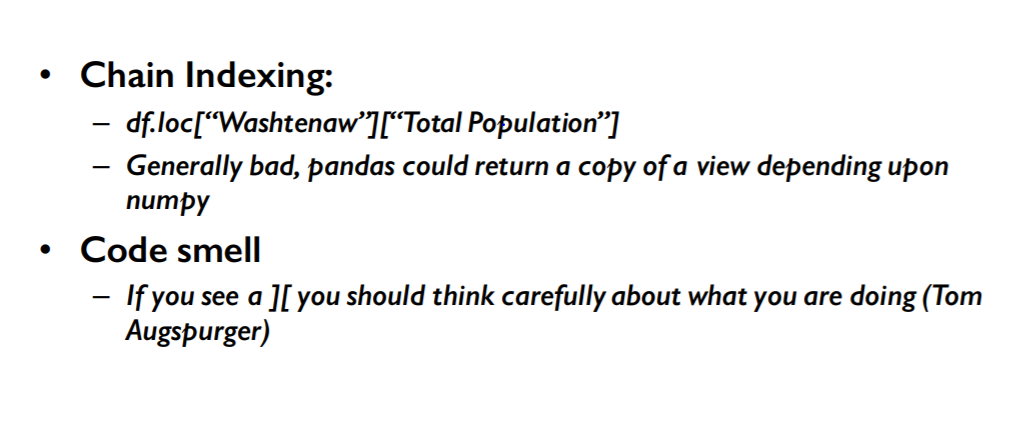
staff\_df

student\_df

pd.merge(staff\_df, student\_df, how='inner', left\_on=['First Name','Last Name'], right\_on=['First Name','Last Name']) #inner join on 2 parts of name



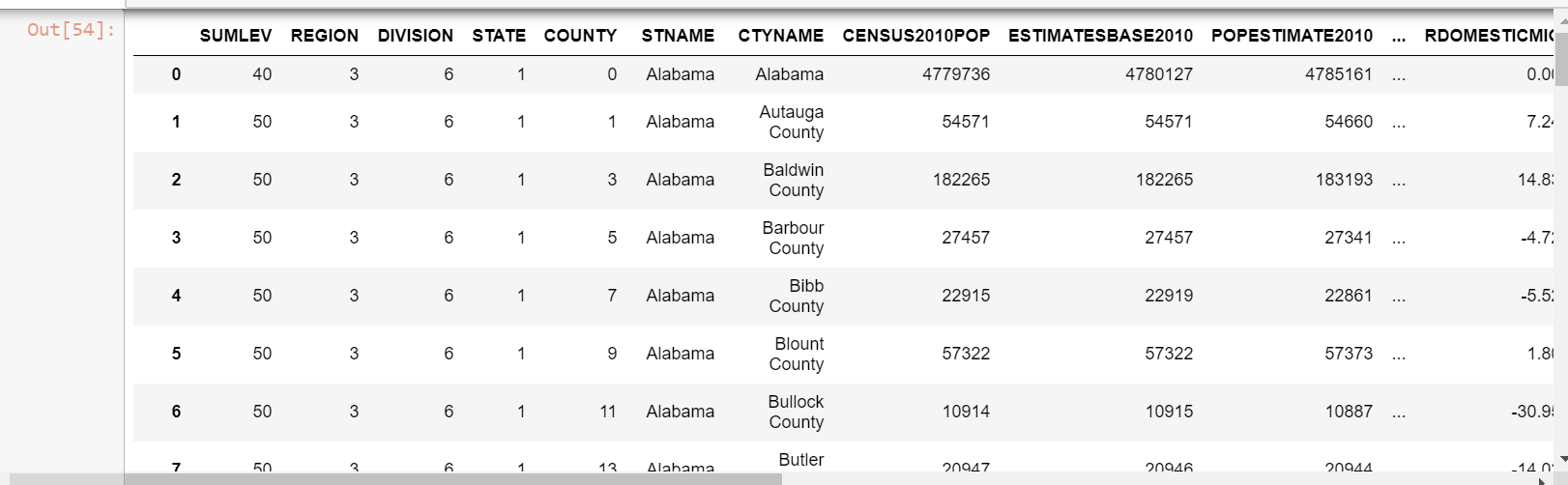
# Idiomatic Pandas: Making Code Pandorable



import pandas as pd

df = pd.read\_csv('census.csv')

df



(df.where(df['SUMLEV']==50)

.dropna()

.set\_index(['STNAME','CTYNAME'])

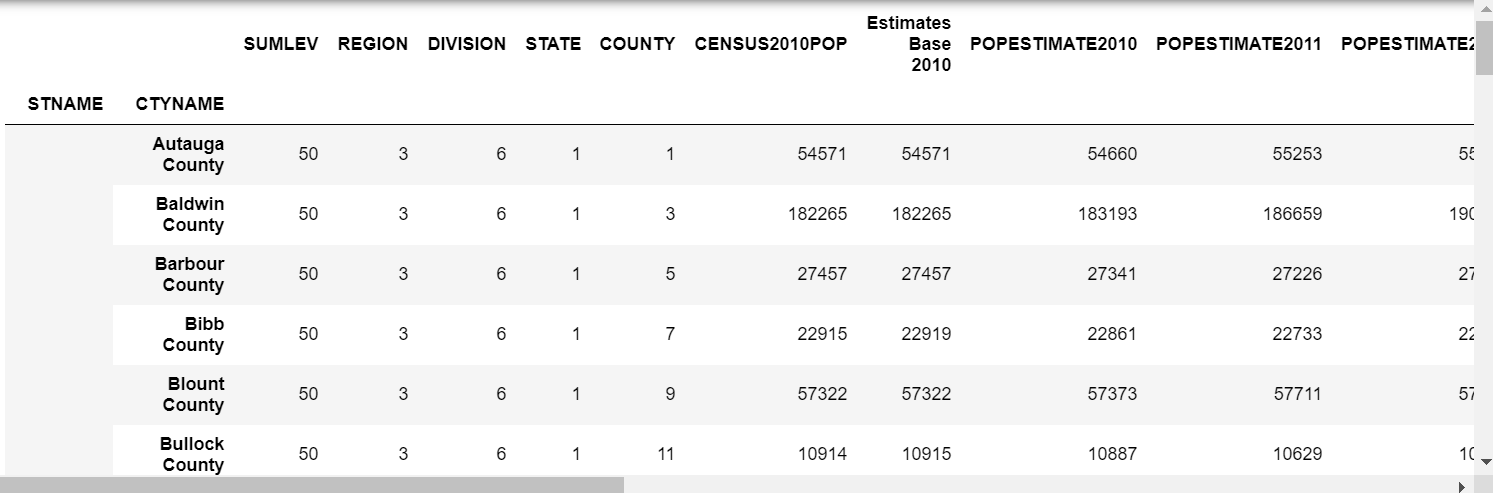
.rename(columns={'ESTIMATESBASE2010': 'Estimates Base 2010'}))

# same as

df = df[df['SUMLEV']==50]

df.set\_index(['STNAME','CTYNAME'], inplace=True)

df.rename(columns={'ESTIMATESBASE2010': 'Estimates Base 2010'})



Example:

print(df.drop(df[df['Quantity'] == 0].index).rename(columns={'Weight': 'Weight (oz.)'}))

pandas has ‘applymap’:

import numpy as np

def min\_max(row):

data = row[['POPESTIMATE2010',

'POPESTIMATE2011',

'POPESTIMATE2012',

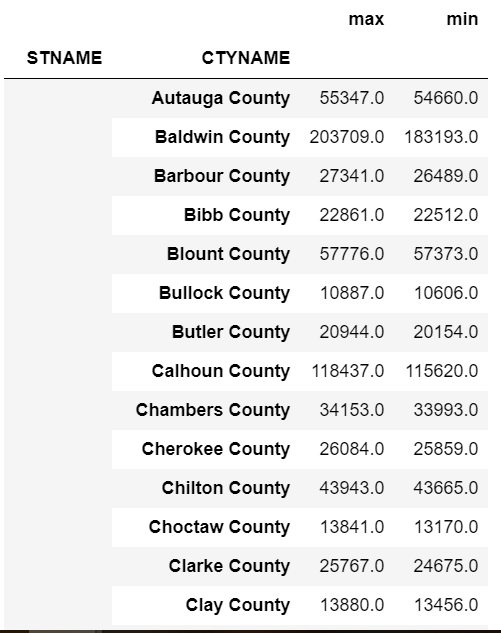
'POPESTIMATE2013',

'POPESTIMATE2014',

'POPESTIMATE2015']]

return pd.Series({'min': np.min(data), 'max': np.max(data)})

df.apply(min\_max, axis=1) #apply min\_max on all rows



import numpy as np

def min\_max(row):

data = row[['POPESTIMATE2010',

'POPESTIMATE2011',

'POPESTIMATE2012',

'POPESTIMATE2013',

'POPESTIMATE2014',

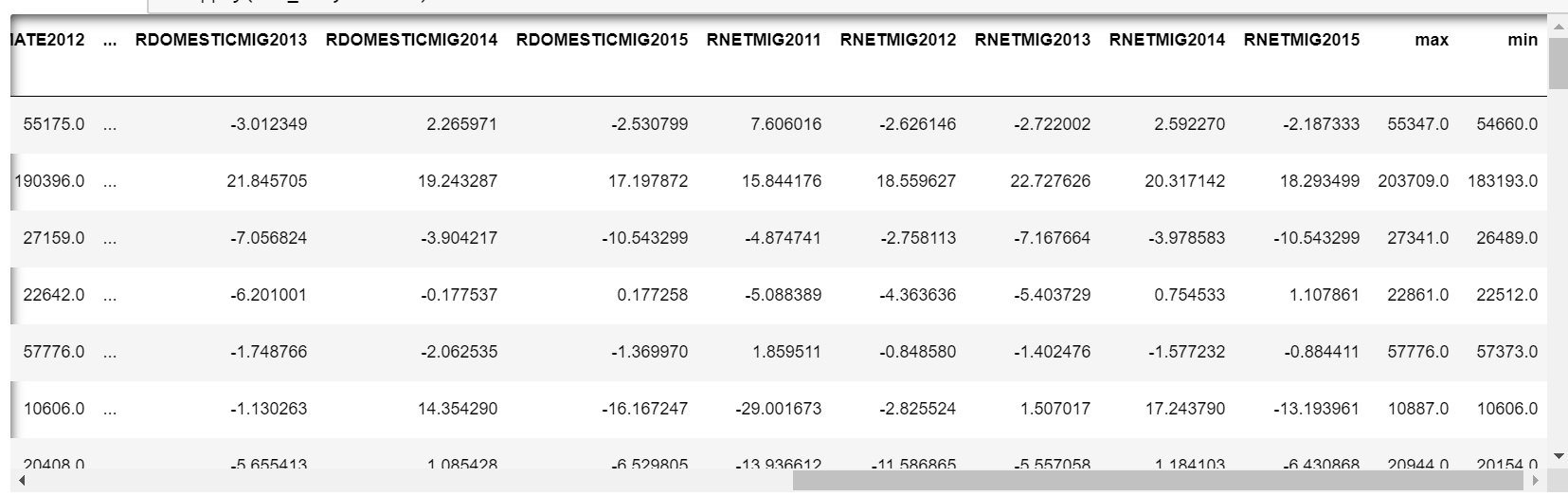
'POPESTIMATE2015']]

row['max'] = np.max(data)

row['min'] = np.min(data)

return row

df.apply(min\_max, axis=1)



rows = ['POPESTIMATE2010',

'POPESTIMATE2011',

'POPESTIMATE2012',

'POPESTIMATE2013',

'POPESTIMATE2014',

'POPESTIMATE2015']

df.apply(lambda x: np.max(x[rows]), axis=1)

STNAME CTYNAME

Alabama Autauga County 55347.0

Baldwin County 203709.0

Barbour County 27341.0

Bibb County 22861.0

Blount County 57776.0

Bullock County 10887.0

Butler County 20944.0

Calhoun County 118437.0

Chambers County 34153.0

Cherokee County 26084.0

Chilton County 43943.0

Choctaw County 13841.0

Clarke County 25767.0

Clay County 13880.0

Cleburne County 15072.0

Coffee County 51211.0

Colbert County 54514.0

Conecuh County 13208.0

Coosa County 11758.0

Covington County 38060.0

Crenshaw County 13963.0

Cullman County 82005.0

Dale County 50358.0

Dallas County 43803.0

DeKalb County 71387.0

Elmore County 81468.0

Escambia County 38309.0

Etowah County 104442.0

Fayette County 17231.0

Franklin County 31734.0

...

Wisconsin Washburn County 15930.0

Washington County 133674.0

Waukesha County 396488.0

Waupaca County 52422.0

Waushara County 24581.0

Winnebago County 169639.0

Wood County 74807.0

Wyoming Albany County 37956.0

Big Horn County 12022.0

Campbell County 49220.0

Carbon County 15856.0

Converse County 14343.0

Crook County 7444.0

Fremont County 41129.0

Goshen County 13666.0

Hot Springs County 4846.0

Johnson County 8636.0

Laramie County 97121.0

Lincoln County 18722.0

Natrona County 82178.0

Niobrara County 2548.0

Park County 29237.0

Platte County 8812.0

Sheridan County 30020.0

Sublette County 10418.0

Sweetwater County 45162.0

Teton County 23125.0

Uinta County 21102.0

Washakie County 8545.0

Weston County 7234.0

dtype: float64

# Group by

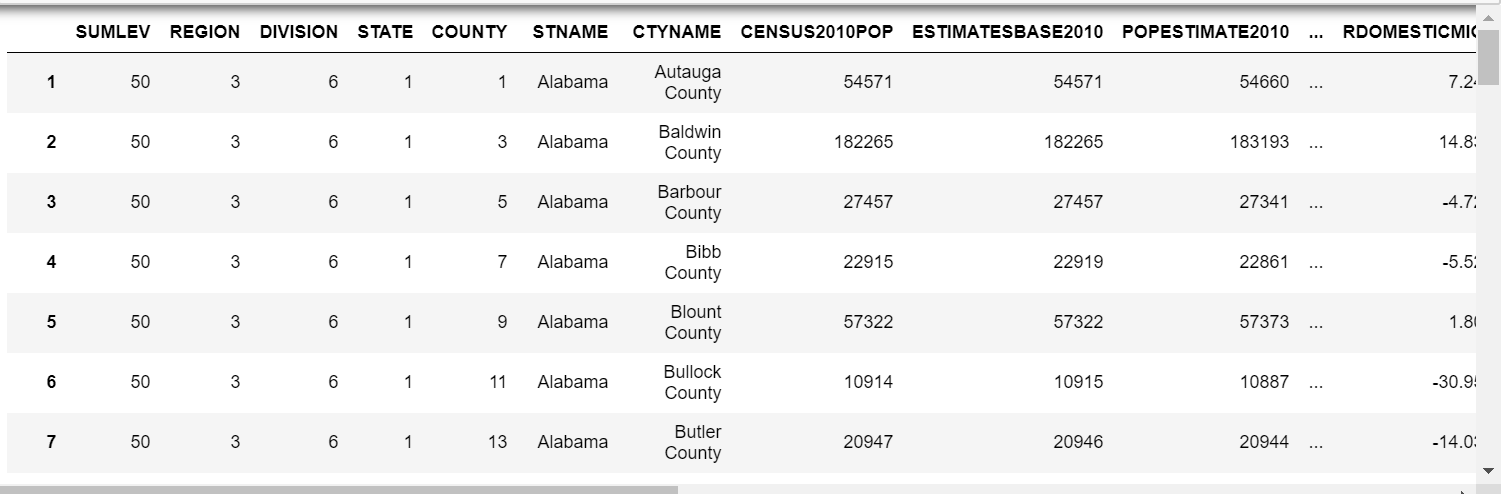
import pandas as pd

import numpy as np

df = pd.read\_csv('census.csv')

df = df[df['SUMLEV']==50]

df



%%timeit -n 10

for state in df['STNAME'].unique():

avg = np.average(df.where(df['STNAME']==state).dropna()['CENSUS2010POP'])

# only pick up 'CENSUS2010POP' and calculate avg for each state

print('Counties in state ' + state + ' have an average population of ' + str(avg))

…

Counties in state Oklahoma have an average population of 48718.8441558

Counties in state Oregon have an average population of 106418.722222

Counties in state Pennsylvania have an average population of 189587.746269

Counties in state Rhode Island have an average population of 210513.4

Counties in state South Carolina have an average population of 100551.391304

Counties in state South Dakota have an average population of 12336.0606061

Counties in state Tennessee have an average population of 66801.1052632

Counties in state Texas have an average population of 98998.2716535

Counties in state Utah have an average population of 95306.3793103

Counties in state Vermont have an average population of 44695.7857143

Counties in state Virginia have an average population of 60111.2932331

Counties in state Washington have an average population of 172424.102564

Counties in state West Virginia have an average population of 33690.8

Counties in state Wisconsin have an average population of 78985.9166667

Counties in state Wyoming have an average population of 24505.4782609

1.5 s ± 14.9 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

%%timeit -n 10

for group, frame in df.**groupby**('STNAME'):

avg = np.average(frame['CENSUS2010POP'])

print('Counties in state ' + group + ' have an average population of ' + str(avg)) # same result

...

Counties in state South Dakota have an average population of 12336.0606061

Counties in state Tennessee have an average population of 66801.1052632

Counties in state Texas have an average population of 98998.2716535

Counties in state Utah have an average population of 95306.3793103

Counties in state Vermont have an average population of 44695.7857143

Counties in state Virginia have an average population of 60111.2932331

Counties in state Washington have an average population of 172424.102564

Counties in state West Virginia have an average population of 33690.8

Counties in state Wisconsin have an average population of 78985.9166667

Counties in state Wyoming have an average population of 24505.4782609

51.2 ms ± 2.64 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)

df = df.set\_index('STNAME')

def fun(item): # item is index

if item[0]<'M': # if the capital letter is before 'M'

return 0

if item[0]<'Q': # if the capital letter is before 'Q'

return 1

return 2

for group, frame in df.groupby(fun):

print('There are ' + str(len(frame)) + ' records in group ' + str(group) + ' for processing.')

There are 1177 records in group 0 for processing.

There are 1134 records in group 1 for processing.

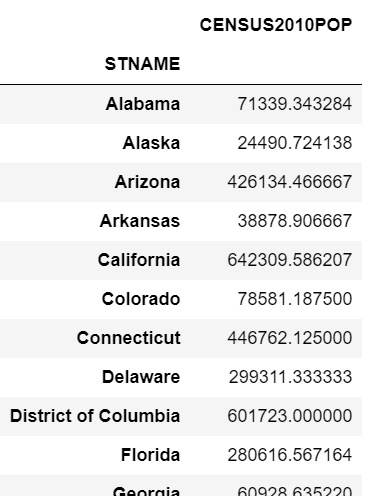
There are 831 records in group 2 for processing.

df = pd.read\_csv('census.csv')

df = df[df['SUMLEV']==50]

df.gniroupby('STNAME').**agg**({'CENSUS2010POP': np.average

})



Example2:

df.groupby('Category').apply(lambda df, a, b: sum(df[a] \* df[b]), 'Weight (oz.)', 'Quantity')

# calculate total weight

print(type(df.groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']))

print(type(df.groupby(level=0)['POPESTIMATE2010']))

<class 'pandas.core.groupby.DataFrameGroupBy'>

<class 'pandas.core.groupby.SeriesGroupBy'>

(df.set\_index('STNAME').groupby(level=0)['CENSUS2010POP']

.agg({'avg': np.average, 'sum': np.sum}))



(df.set\_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']

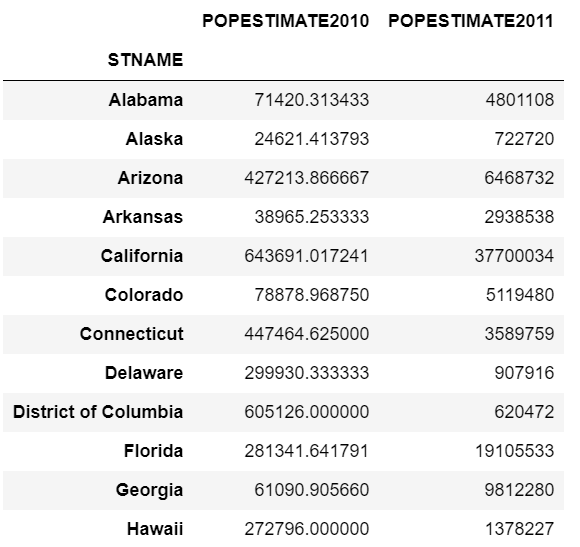
.agg({'avg': np.average, 'sum': np.sum}))



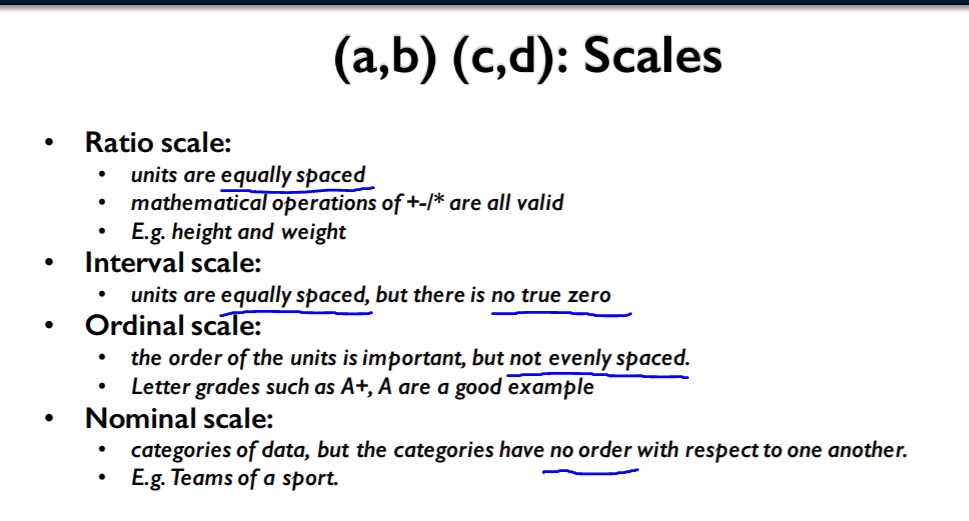
(df.set\_index('STNAME').groupby(level=0)['POPESTIMATE2010','POPESTIMATE2011']

.agg({'POPESTIMATE2010': np.average, 'POPESTIMATE2011': np.sum}))

# 如果新column的agg为对应原名，仅针对对应的column



# Scales

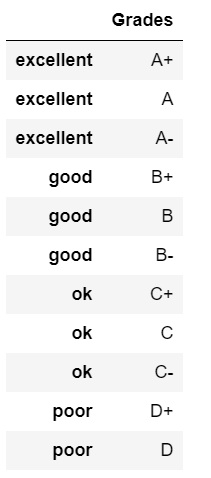


df = pd.DataFrame(['A+', 'A', 'A-', 'B+', 'B', 'B-', 'C+', 'C', 'C-', 'D+', 'D'],

index=['excellent', 'excellent', 'excellent', 'good', 'good', 'good', 'ok', 'ok', 'ok', 'poor', 'poor'])

df.rename(**columns={0: 'Grades'}**, inplace=True)

df



df['Grades'].astype('category').head() # set type into category

excellent A+

excellent A

excellent A-

good B+

good B

Name: Grades, dtype: category

Categories (11, object): [A, A+, A-, B, ..., C+, C-, D, D+]

grades = df['Grades'].astype('category',

categories=['D', 'D+', 'C-', 'C', 'C+', 'B-', 'B', 'B+', 'A-', 'A', 'A+'],

ordered=True) # set type into category and make an order

grades.head()

excellent A+

excellent A

excellent A-

good B+

good B

Name: Grades, dtype: category

Categories (11, object): [D < D+ < C- < C ... B+ < A- < A < A+]

grades > 'C'

excellent True

excellent True

excellent True

good True

good True

good True

ok True

ok False

ok False

poor False

poor False

Name: Grades, dtype: bool

Example3:

s = pd.Series(['Low', 'Low', 'High', 'Medium', 'Low', 'High', 'Low'])

s.astype('category', categories=['Low', 'Medium', 'High'], ordered=True)

df = pd.read\_csv('census.csv')

df = df[df['SUMLEV']==50]

df = df.set\_index('STNAME').groupby(level=0)['CENSUS2010POP'].agg({'avg': np.average})

pd.cut(df['avg'],10) # slice into 10 groups and divide them into those groups

STNAME

Alabama (11706.0871, 75333.413]

Alaska (11706.0871, 75333.413]

Arizona (390320.176, 453317.529]

Arkansas (11706.0871, 75333.413]

California (579312.234, 642309.586]

Colorado (75333.413, 138330.766]

Connecticut (390320.176, 453317.529]

Delaware (264325.471, 327322.823]

District of Columbia (579312.234, 642309.586]

Florida (264325.471, 327322.823]

Georgia (11706.0871, 75333.413]

Hawaii (264325.471, 327322.823]

Idaho (11706.0871, 75333.413]

Illinois (75333.413, 138330.766]

Indiana (11706.0871, 75333.413]

Iowa (11706.0871, 75333.413]

Kansas (11706.0871, 75333.413]

Kentucky (11706.0871, 75333.413]

Louisiana (11706.0871, 75333.413]

Maine (75333.413, 138330.766]

Maryland (201328.118, 264325.471]

Massachusetts (453317.529, 516314.881]

Michigan (75333.413, 138330.766]

Minnesota (11706.0871, 75333.413]

Mississippi (11706.0871, 75333.413]

Missouri (11706.0871, 75333.413]

Montana (11706.0871, 75333.413]

Nebraska (11706.0871, 75333.413]

Nevada (138330.766, 201328.118]

New Hampshire (75333.413, 138330.766]

New Jersey (390320.176, 453317.529]

New Mexico (11706.0871, 75333.413]

New York (264325.471, 327322.823]

North Carolina (75333.413, 138330.766]

North Dakota (11706.0871, 75333.413]

Ohio (75333.413, 138330.766]

Oklahoma (11706.0871, 75333.413]

Oregon (75333.413, 138330.766]

Pennsylvania (138330.766, 201328.118]

Rhode Island (201328.118, 264325.471]

South Carolina (75333.413, 138330.766]

South Dakota (11706.0871, 75333.413]

Tennessee (11706.0871, 75333.413]

Texas (75333.413, 138330.766]

Utah (75333.413, 138330.766]

Vermont (11706.0871, 75333.413]

Virginia (11706.0871, 75333.413]

Washington (138330.766, 201328.118]

West Virginia (11706.0871, 75333.413]

Wisconsin (75333.413, 138330.766]

Wyoming (11706.0871, 75333.413]

Name: avg, dtype: category

Categories (10, object): [(11706.0871, 75333.413] < (75333.413, 138330.766] < (138330.766, 201328.118] < (201328.118, 264325.471] ... (390320.176, 453317.529] < (453317.529, 516314.881] < (516314.881, 579312.234] < (579312.234, 642309.586]]

Example4:

s = pd.Series([168, 180, 174, 190, 170, 185, 179, 181, 175, 169, 182, 177, 180, 171])

pd.cut(s, 3) # cut into 3 groups

# Pivot Tables

df = pd.DataFrame({"A": ["foo", "foo", "foo", "foo", "foo",

"bar", "bar", "bar", "bar"],

"B": ["one", "one", "one", "two", "two",

"one", "one", "two", "two"],

"C": ["small", "large", "large", "small",

"small", "large", "small", "small",

"large"],

"D": [1, 2, 2, 3, 3, 4, 5, 6, 7],

"E": [2, 4, 5, 5, 6, 6, 8, 9, 9]})

df



table = pd.**pivot\_table**(df, values='D', index=['A', 'B'],

columns=['C'], aggfunc=np.sum)

table

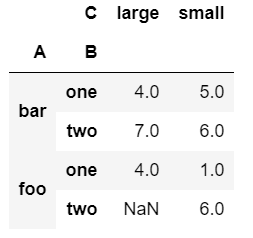
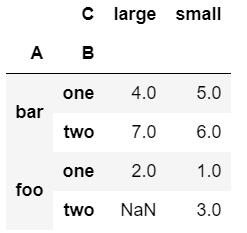


table = pd.pivot\_table(df, values='D', index=['A', 'B'],

columns=['C'], aggfunc=np.mean)

table



#http://open.canada.ca/data/en/dataset/98f1a129-f628-4ce4-b24d-6f16bf24dd64

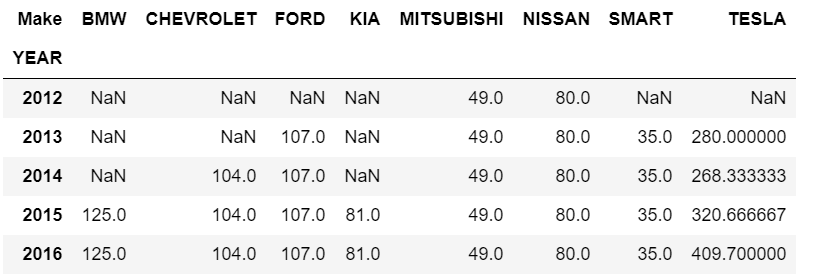
df = pd.read\_csv('cars.csv')

df.head()

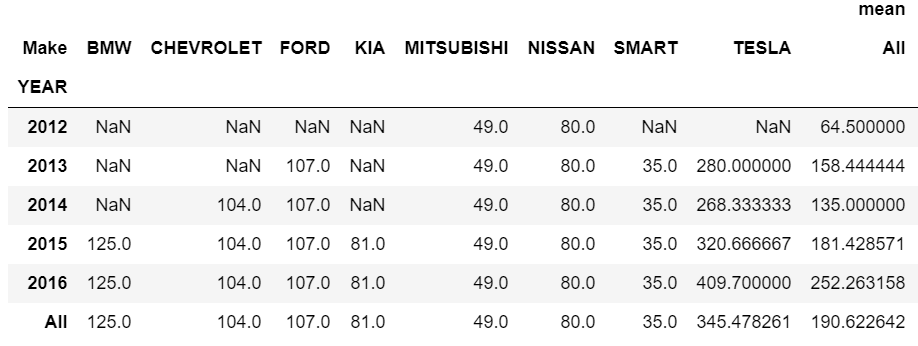


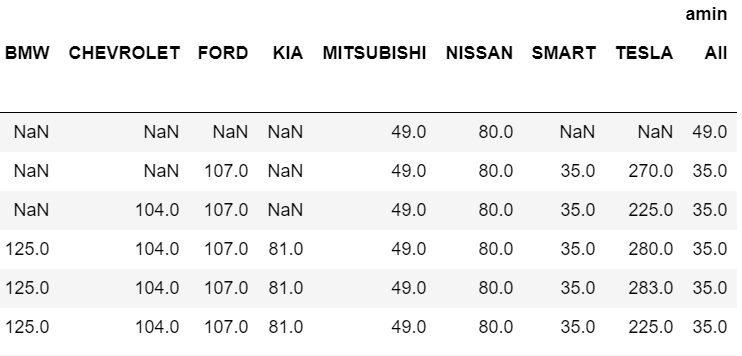
df.pivot\_table(values='(kW)', index='YEAR', columns='Make', aggfunc=np.mean)

# choose KW, order by year, separate by which company makes it, get means



df.pivot\_table(values='(kW)', index='YEAR', columns='Make', aggfunc=[np.mean,np.min], margins=True) # add min, and summarize subtotal/total





Example5:

pd.pivot\_table(Bikes, index=['Manufacturer','Bike Type'])

# Date Functionality in Pandas

import pandas as pd

import numpy as np

### Timestamp

pd.**Timestamp**('9/1/2016 10:05AM')

Timestamp('2016-09-01 10:05:00')

### Period

pd.**Period**('1/2016')

Period('2016-01', 'M')

pd.**Period**('3/5/2016')

Period('2016-03-05', 'D')

### DatetimeIndex

t1 = pd.Series(list('abc'), [pd.Timestamp('2016-09-01'), pd.Timestamp('2016-09-02'), pd.Timestamp('2016-09-03')])

t1

2016-09-01 a

2016-09-02 b

2016-09-03 c

dtype: object

type(t1.index)

pandas.tseries.index.DatetimeIndex

### PeriodIndex

t2 = pd.Series(list('def'), [pd.Period('2016-09'), pd.Period('2016-10'), pd.Period('2016-11')])

t2

2016-09 d

2016-10 e

2016-11 f

Freq: M, dtype: object

type(t2.index)

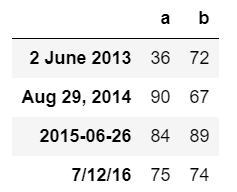
pandas.tseries.period.PeriodIndex

### Converting to Datetime

d1 = ['2 June 2013', 'Aug 29, 2014', '2015-06-26', '7/12/16']

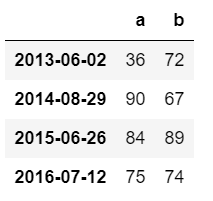
ts3 = pd.DataFrame(np.random.randint(10, 100, (4,2)), index=d1, columns=list('ab'))

ts3



ts3.index = pd.to\_datetime(ts3.index)

ts3



pd.to\_datetime('4.7.12', dayfirst=True)

Timestamp('2012-07-04 00:00:00')

### Timedeltas

pd.Timestamp('9/3/2016')-pd.Timestamp('9/1/2016')

Timedelta('2 days 00:00:00')

pd.Timestamp('9/2/2016 8:10AM') + pd.Timedelta('12D 3H')

Timestamp('2016-09-14 11:10:00')

### Working with Dates in a Dataframe

dates = pd.**date\_range**('10-01-2016', periods=9, freq='2W-SUN')

dates

DatetimeIndex(['2016-10-02', '2016-10-16', '2016-10-30', '2016-11-13',

'2016-11-27', '2016-12-11', '2016-12-25', '2017-01-08',

'2017-01-22'],

dtype='datetime64[ns]', freq='2W-SUN')

l = np.random.randint(-5, 10, 9)

l

array([ 9, 8, -4, 3, 8, 3, -1, 1, -3])

l**.cumsum**()

array([ 9, 17, 13, 16, 24, 27, 26, 27, 24])

df = pd.DataFrame({'Count 1': 100 + np.random.randint(-5, 10, 9).cumsum(),

'Count 2': 120 + np.random.randint(-5, 10, 9)}, index=dates)

df



df.index.**weekday\_name**

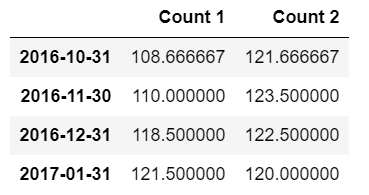
array(['Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunday', 'Sunday',

'Sunday', 'Sunday', 'Sunday'], dtype=object)

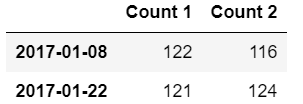
df.diff()



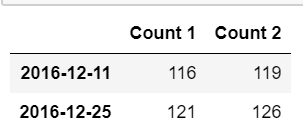
df.**resample**('M').mean()

\

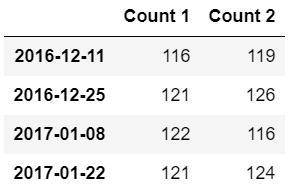
df['2017'] # only year 2017



df['2016-12']



df['2016-12':] # after ‘2016-12’



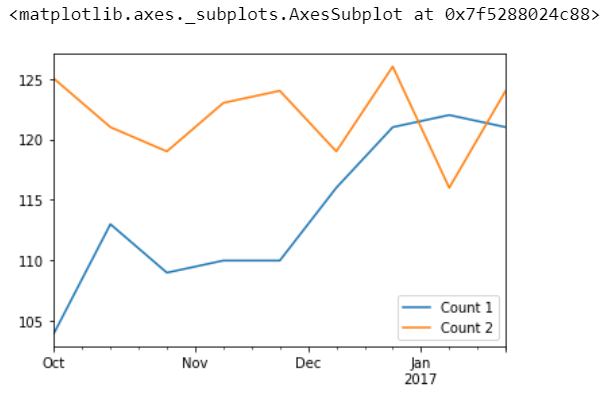
df.asfreq('W', method='ffill') # week, forward fill



**import matplotlib.pyplot as plt**

**%matplotlib inline**

**df.plot()**



Skills:

def answer\_eleven():

Top15 = answer\_one()

Top15['PredPopu'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']

ContinentDict = {'China':'Asia',

'United States':'North America',

'Japan':'Asia',

'United Kingdom':'Europe',

'Russian Federation':'Europe',

'Canada':'North America',

'Germany':'Europe',

'India':'Asia',

'France':'Europe',

'South Korea':'Asia',

'Italy':'Europe',

'Spain':'Europe',

'Iran':'Asia',

'Australia':'Australia',

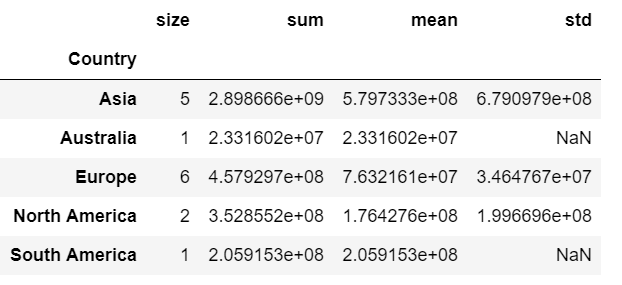
'Brazil':'South America'}

Top15.rename(index=ContinentDict,inplace=True)

df11 = Top15.groupby(level=0)['PredPopu'].agg(['size', 'sum', 'mean', 'std'])

return df11

answer\_eleven()



千分符：

def answer\_thirteen():

Top15 = answer\_one()

Top15['PredPopu'] = Top15['Energy Supply']/Top15['Energy Supply per Capita']

Top15['PredPopu'] =Top15['PredPopu'].apply(lambda x: format(x,','))

return Top15['PredPopu']

answer\_thirteen()