Python For Analysis 14.4~14.6

2017010715 허지혜

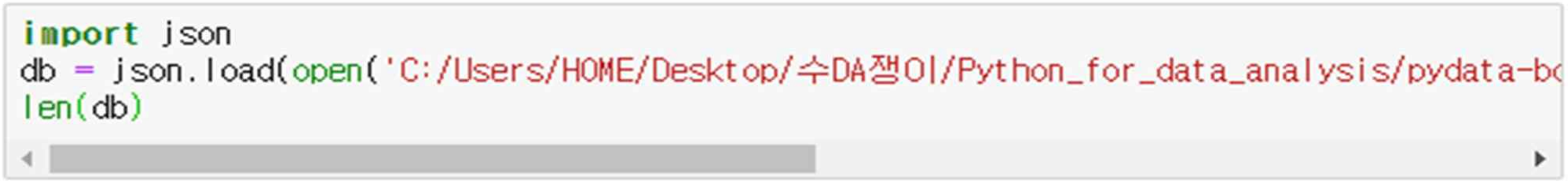
목차

1. 미국농무부 영양소 정보

JSON 파일

```
{ "id": 21441, "description": "KENTUCKY FRIED CHICKEN, Fried Chicken, EXTRA CRISPY,
Wing, meat and skin with breading", "tags": ["KFC"], "manufacturer": "Kentucky Fried Chicken",
"group": "Fast Foods", "portions": [ { "amount": 1, "unit": "wing, with skin", "grams": 68.0 },

      ...

], "nutrients": [ { "value": 20.8, "units": "g", "description": "Protein", "group": "Composition" },

      ...

]}
```

=> JSON 파일은 분석하기 편하지 않으므로 더 나은 형태로 바꿔보자 !



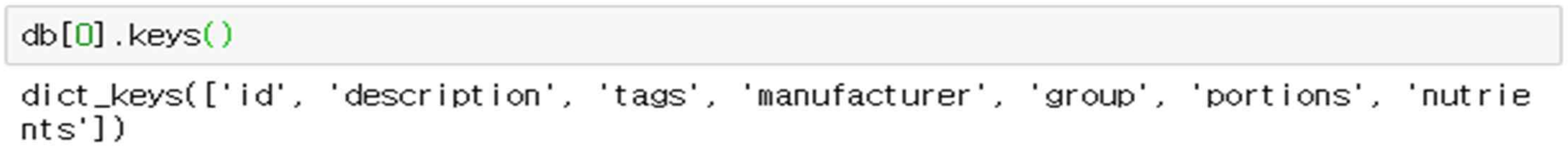
```python
import json
db = json.load(open('C:/Users/HOME/Desktop/수DA쟁이/Python_for_data_analysis/pydata-bo
len(db)
```

```
6636
```

각 음식은 숫자로 된 고유 ID 뿐 아니라 영양소와 제공량 등 두 리스트를 가지고 있다.

1. 미국농무부 영양소 정보

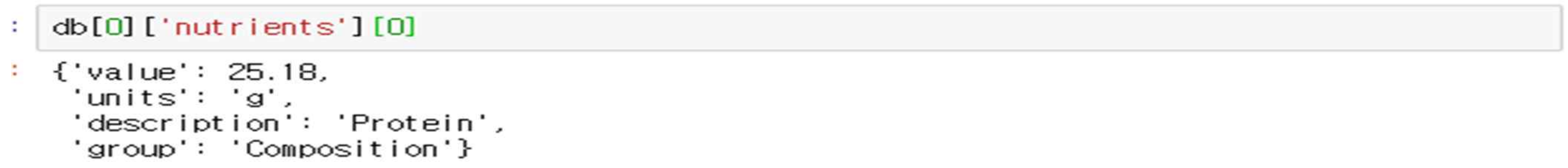
Db에 있는 각 엔트리는 한 가지 음식에 대한 모든 정보를 담고 있는 사전형이다.



```
db[0].keys()
```

```
dict_keys(['id', 'description', 'tags', 'manufacturer', 'group', 'portions', 'nutrie
nts'])
```

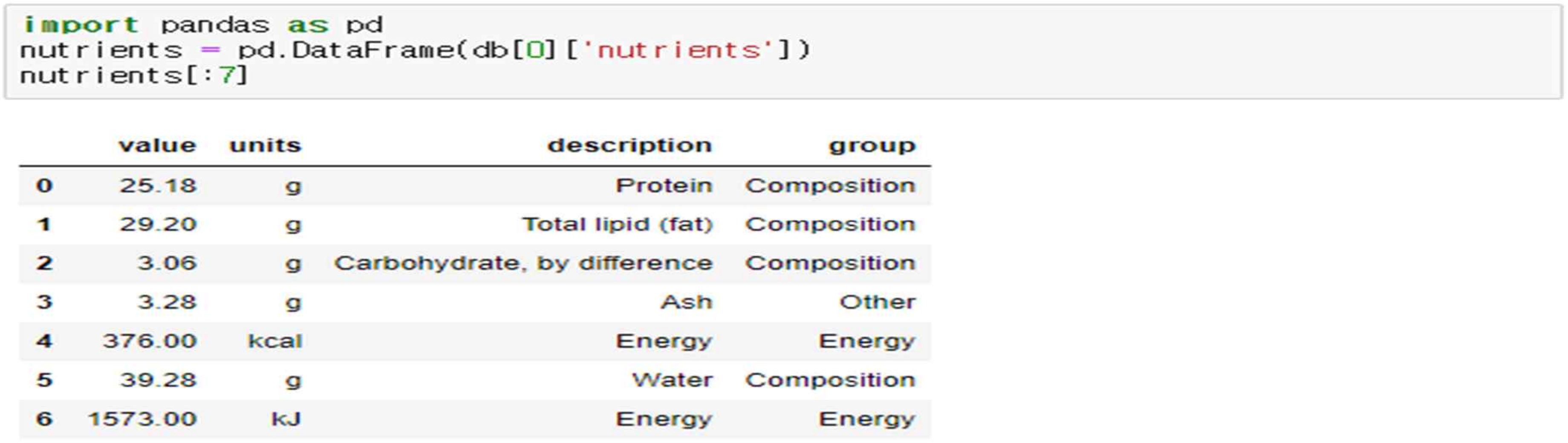
Nutrients는 사전의 리스트이며 각 항목은 한 가지 영양소에 대한 정보는 담고 있다.



```
: db[0]['nutrients'][0]
```

```
: {'value': 25.18,
   'units': 'g',
   'description': 'Protein',
   'group': 'Composition'}
```

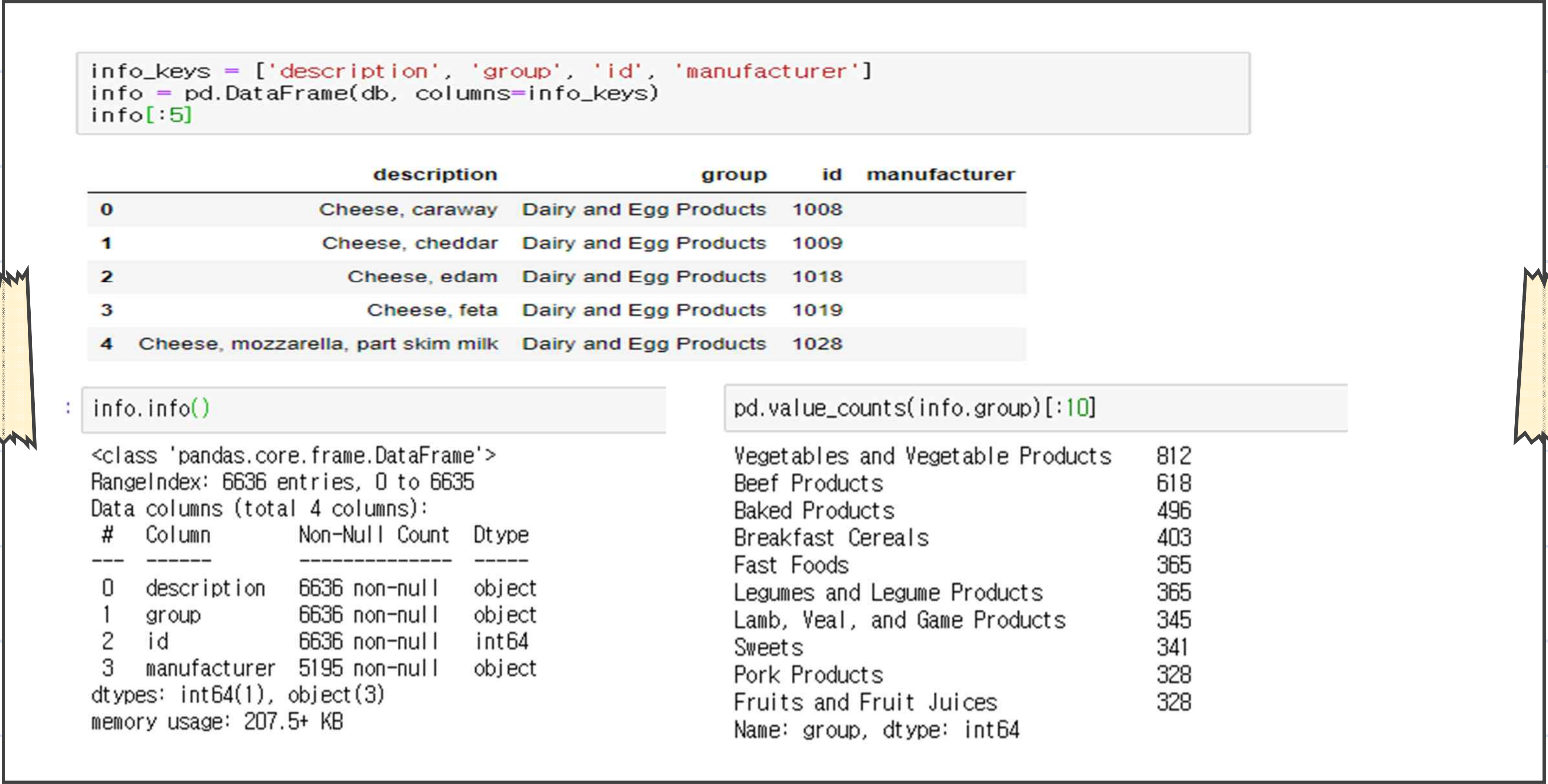
1. 미국농무부 영양소 정보



```
import pandas as pd
nutrients = pd.DataFrame(db[0]['nutrients'])
nutrients[:7]
```

	value	units	description	group
0	25.18	g	Protein	Composition
1	29.20	g	Total lipid (fat)	Composition
2	3.06	g	Carbohydrate, by difference	Composition
3	3.28	g	Ash	Other
4	376.00	kcal	Energy	Energy
5	39.28	g	Water	Composition
6	1573.00	kJ	Energy	Energy

1. 미국농무부 영양소 정보



```
info_keys = ['description', 'group', 'id', 'manufacturer']
info = pd.DataFrame(db, columns=info_keys)
info[:5]
```

	description	group	id	manufacturer
0	Cheese, caraway	Dairy and Egg Products	1008	
1	Cheese, cheddar	Dairy and Egg Products	1009	
2	Cheese, edam	Dairy and Egg Products	1018	
3	Cheese, feta	Dairy and Egg Products	1019	
4	Cheese, mozzarella, part skim milk	Dairy and Egg Products	1028	

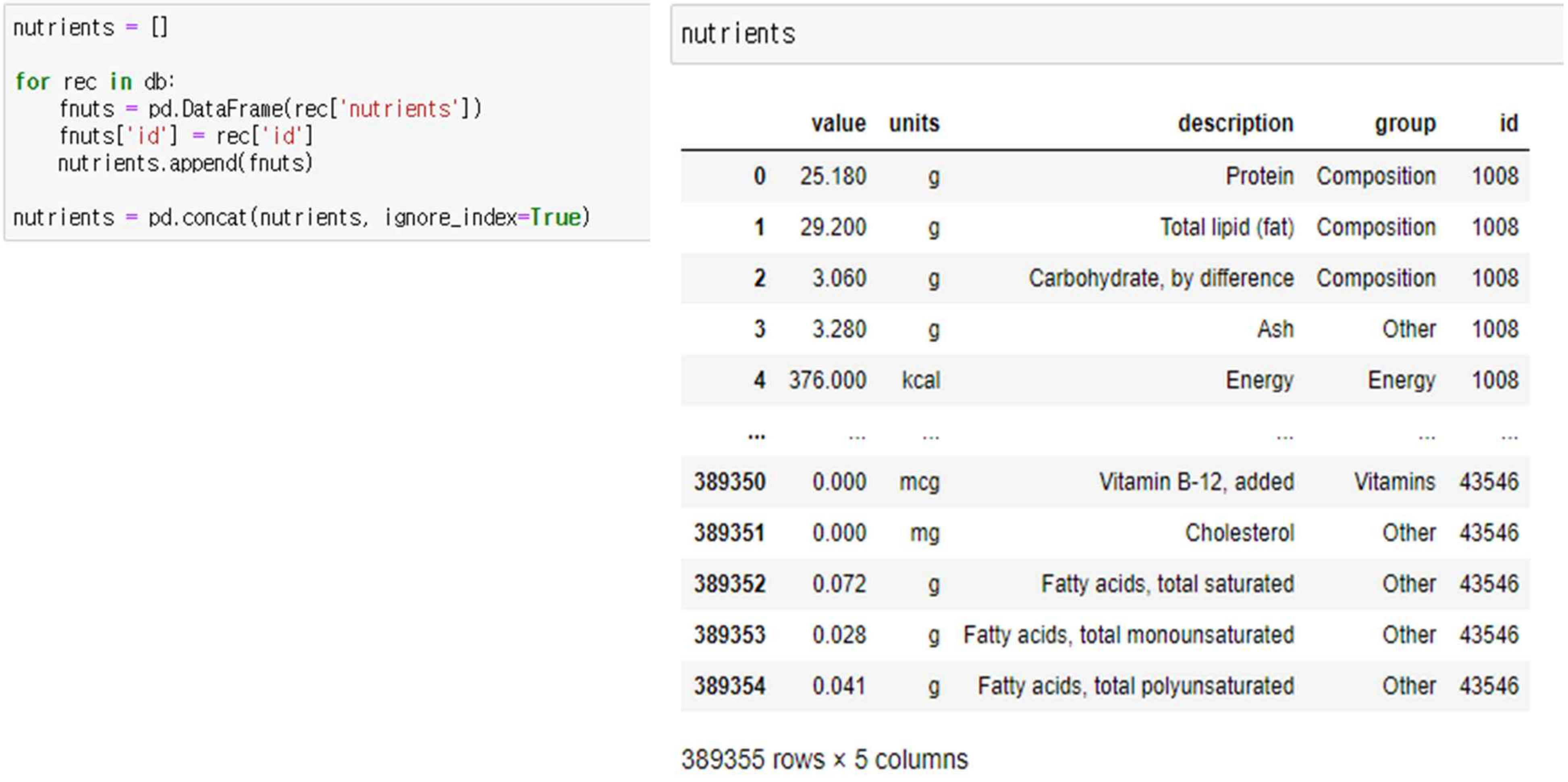
```
info.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6636 entries, 0 to 6635
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   description   6636 non-null   object
 1   group         6636 non-null   object
 2   id            6636 non-null   int64
 3   manufacturer  5195 non-null   object
dtypes: int64(1), object(3)
memory usage: 207.5+ KB
```

```
pd.value_counts(info.group)[:10]
```

```
Vegetables and Vegetable Products    812
Beef Products                        618
Baked Products                       496
Breakfast Cereals                    403
Fast Foods                           365
Legumes and Legume Products          365
Lamb, Veal, and Game Products        345
Sweets                               341
Pork Products                        328
Fruits and Fruit Juices              328
Name: group, dtype: int64
```

1. 미국농무부 영양소 정보

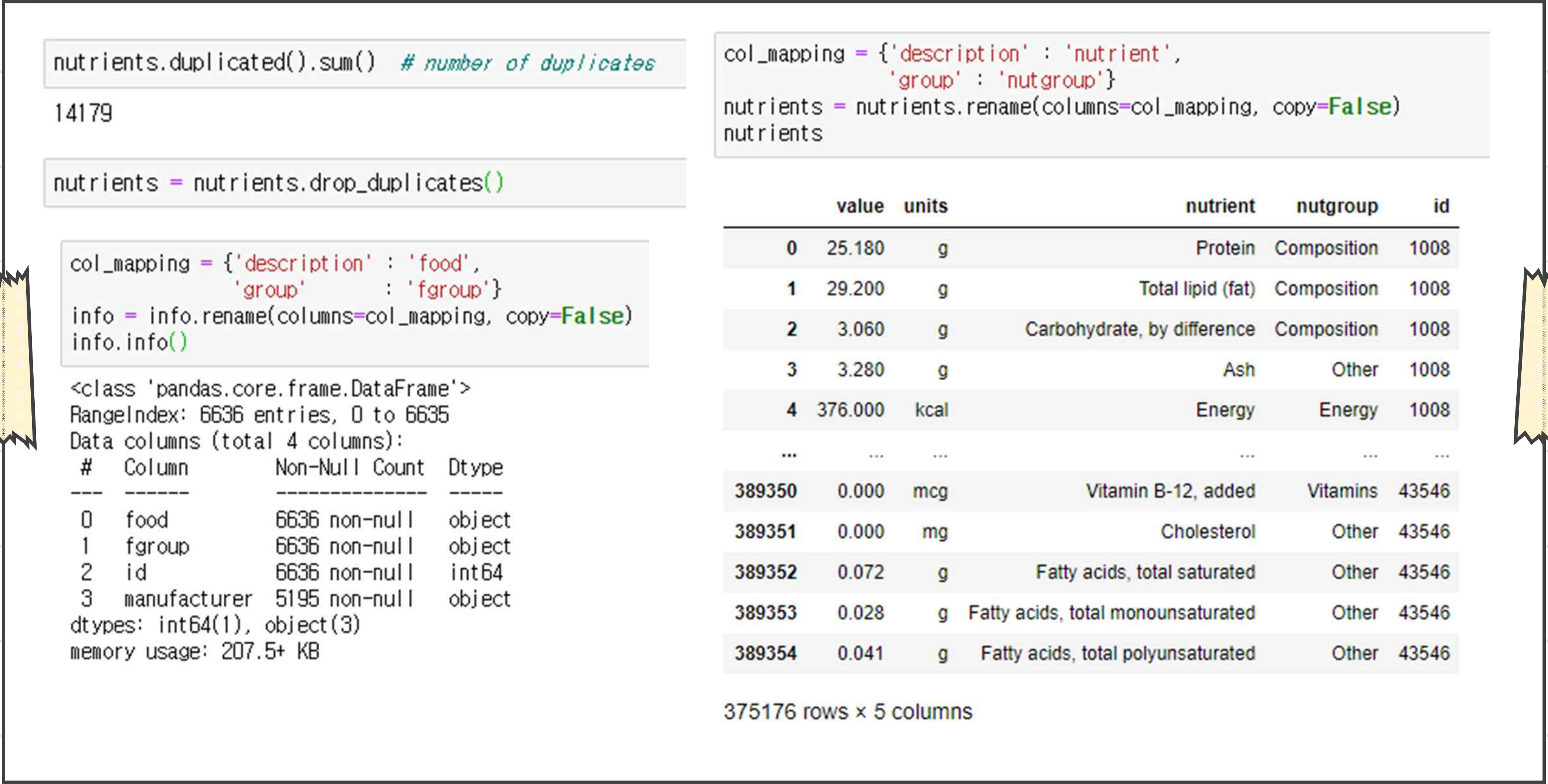


```
nutrients = []

for rec in db:
    fnuts = pd.DataFrame(rec['nutrients'])
    fnuts['id'] = rec['id']
    nutrients.append(fnuts)

nutrients = pd.concat(nutrients, ignore_index=True)
```

```
nutrients
```

```
          value  units                       description        group     id
     0   25.180      g                           Protein  Composition   1008
     1   29.200      g                  Total lipid (fat)  Composition   1008
     2    3.060      g          Carbohydrate, by difference  Composition   1008
     3    3.280      g                               Ash        Other   1008
     4  376.000   kcal                            Energy       Energy   1008
   ...      ...    ...                               ...          ...    ...
389350    0.000    mcg               Vitamin B-12, added     Vitamins  43546
389351    0.000     mg                       Cholesterol        Other  43546
389352    0.072      g         Fatty acids, total saturated     Other  43546
389353    0.028      g  Fatty acids, total monounsaturated     Other  43546
389354    0.041      g  Fatty acids, total polyunsaturated     Other  43546

389355 rows × 5 columns
```

1. 미국농무부 영양소 정보



```
nutrients.duplicated().sum()  # number of duplicates
```

```
14179
```

```
nutrients = nutrients.drop_duplicates()
```

```
col_mapping = {'description' : 'food',
               'group'       : 'fgroup'}
info = info.rename(columns=col_mapping, copy=False)
info.info()
```

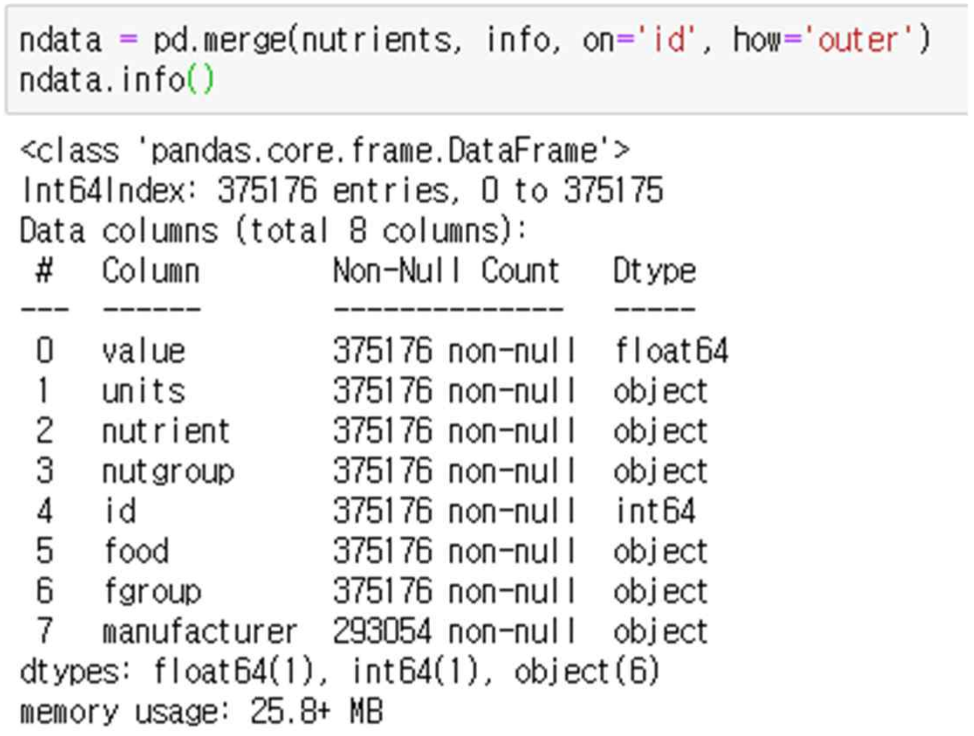
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6636 entries, 0 to 6635
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   food          6636 non-null   object
 1   fgroup        6636 non-null   object
 2   id            6636 non-null   int64 
 3   manufacturer  5195 non-null   object
dtypes: int64(1), object(3)
memory usage: 207.5+ KB
```

```
col_mapping = {'description' : 'nutrient',
               'group' : 'nutgroup'}
nutrients = nutrients.rename(columns=col_mapping, copy=False)
nutrients
```

	value	units	nutrient	nutgroup	id
0	25.180	g	Protein	Composition	1008
1	29.200	g	Total lipid (fat)	Composition	1008
2	3.060	g	Carbohydrate, by difference	Composition	1008
3	3.280	g	Ash	Other	1008
4	376.000	kcal	Energy	Energy	1008
...	...	...	...	...	...
389350	0.000	mcg	Vitamin B-12, added	Vitamins	43546
389351	0.000	mg	Cholesterol	Other	43546
389352	0.072	g	Fatty acids, total saturated	Other	43546
389353	0.028	g	Fatty acids, total monounsaturated	Other	43546
389354	0.041	g	Fatty acids, total polyunsaturated	Other	43546

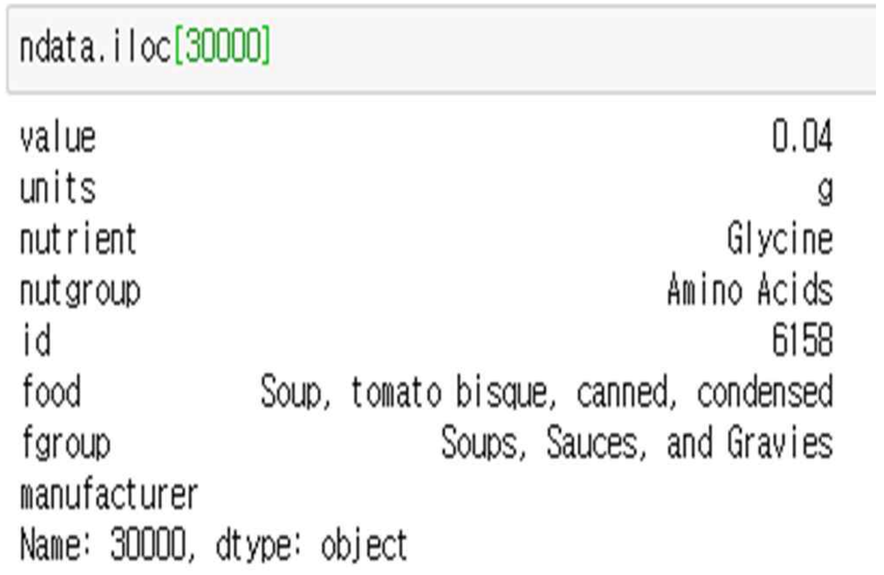
375176 rows × 5 columns

1. 미국농무부 영양소 정보



```
ndata = pd.merge(nutrients, info, on='id', how='outer')
ndata.info()
```

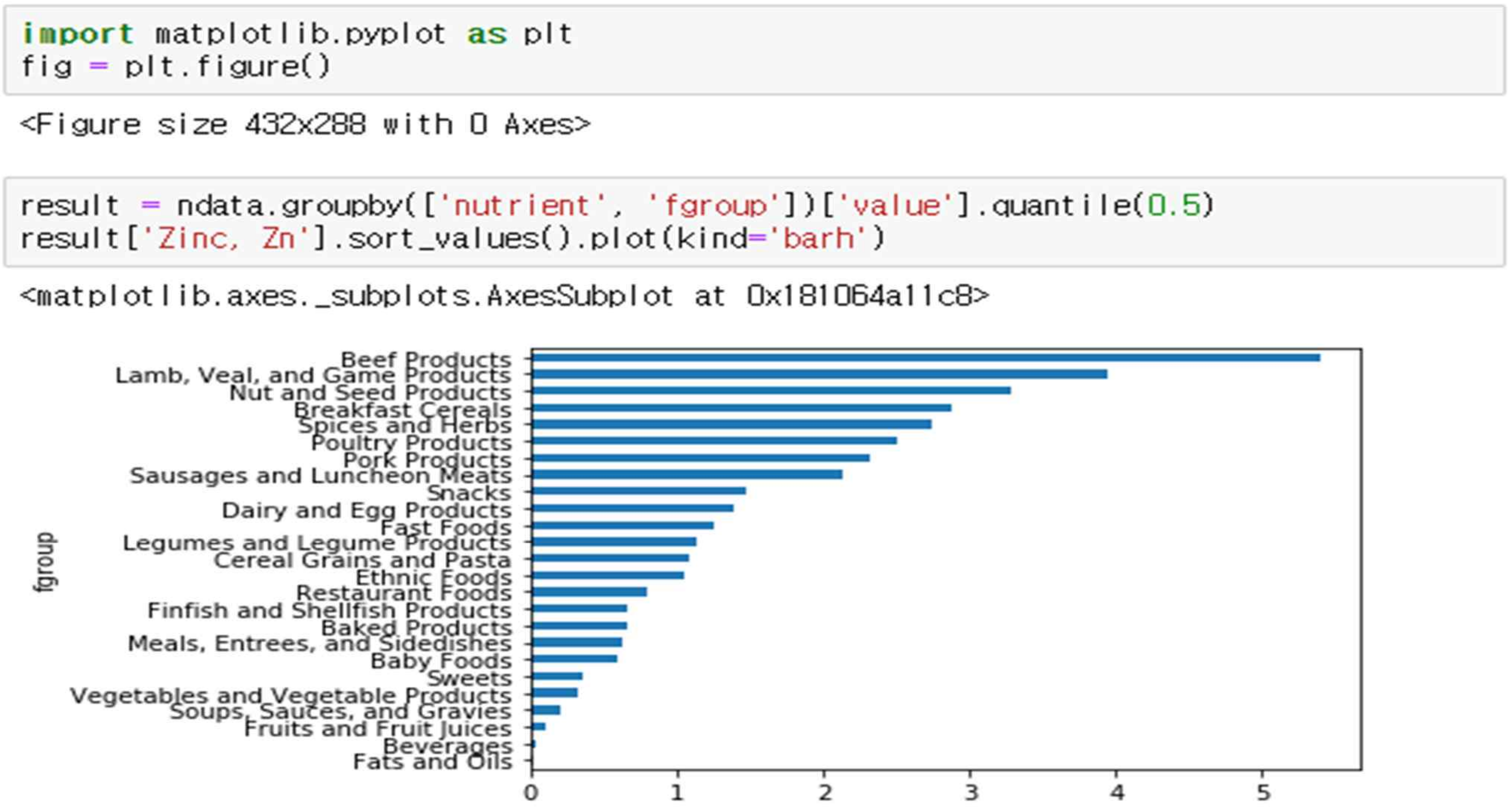
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 375176 entries, 0 to 375175
Data columns (total 8 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   value         375176 non-null  float64
 1   units         375176 non-null  object
 2   nutrient      375176 non-null  object
 3   nutgroup      375176 non-null  object
 4   id            375176 non-null  int64
 5   food          375176 non-null  object
 6   fgroup        375176 non-null  object
 7   manufacturer  293054 non-null  object
dtypes: float64(1), int64(1), object(6)
memory usage: 25.8+ MB
```



```
ndata.iloc[30000]
```

```
value                                              0.04
units                                                 g
nutrient                                        Glycine
nutgroup                                    Amino Acids
id                                                 6158
food            Soup, tomato bisque, canned, condensed
fgroup                         Soups, Sauces, and Gravies
manufacturer
Name: 30000, dtype: object
```

1. 미국농무부 영양소 정보



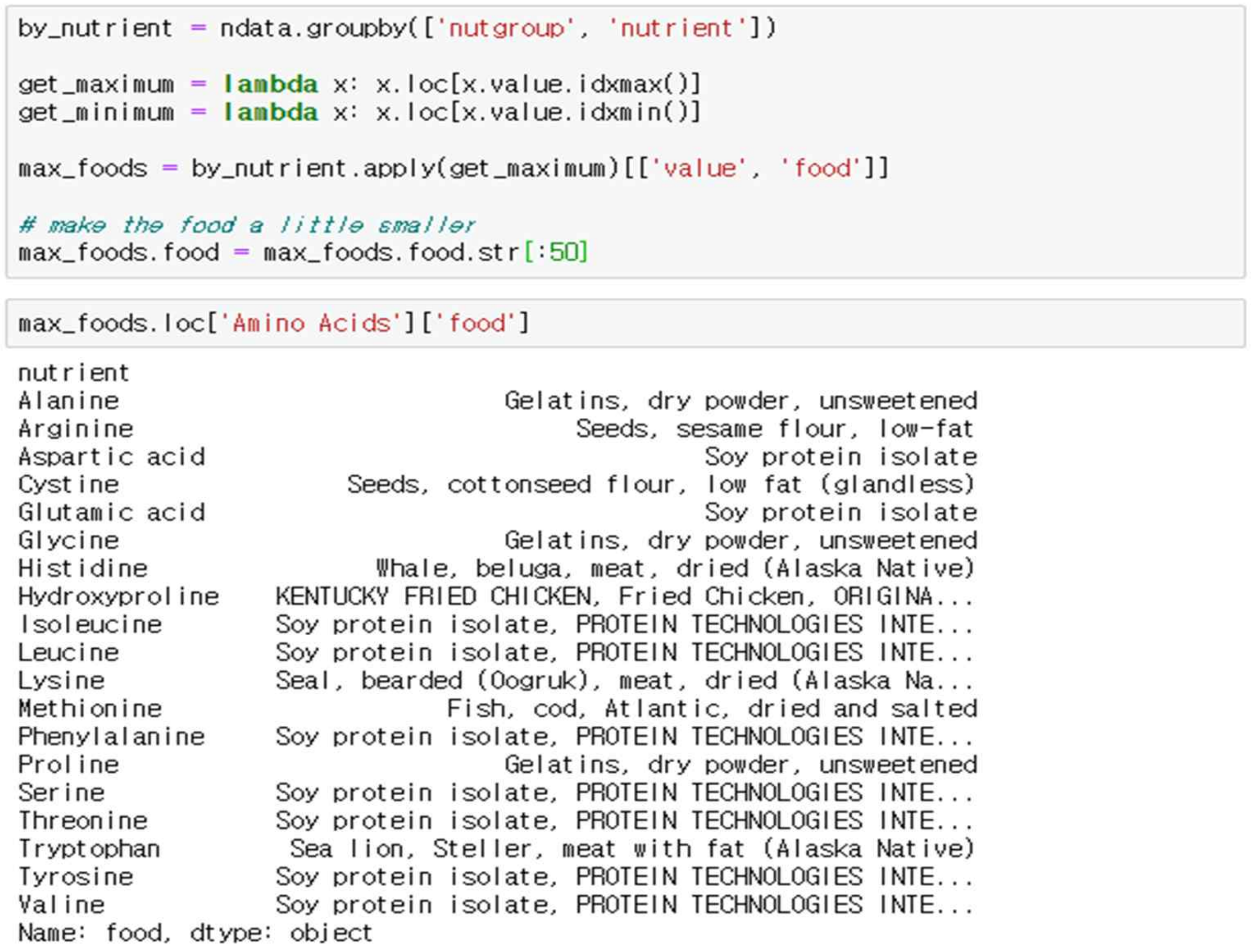
```
import matplotlib.pyplot as plt
fig = plt.figure()
```

```
<Figure size 432x288 with 0 Axes>
```

```
result = ndata.groupby(['nutrient', 'fgroup'])['value'].quantile(0.5)
result['Zinc, Zn'].sort_values().plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x181064a11c8>
```

1. 미국농무부 영양소 정보



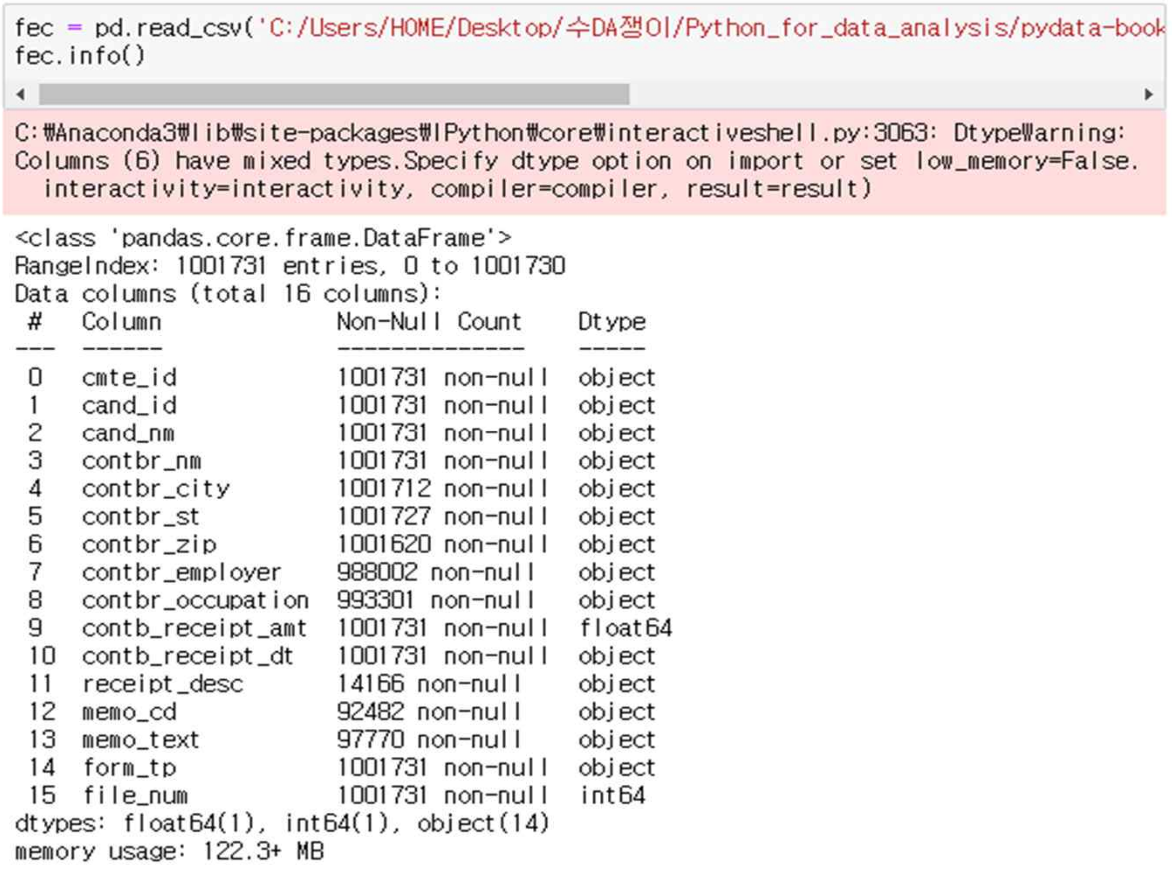
```
by_nutrient = ndata.groupby(['nutgroup', 'nutrient'])

get_maximum = lambda x: x.loc[x.value.idxmax()]
get_minimum = lambda x: x.loc[x.value.idxmin()]

max_foods = by_nutrient.apply(get_maximum)[['value', 'food']]

# make the food a little smaller
max_foods.food = max_foods.food.str[:50]
```

```
max_foods.loc['Amino Acids']['food']
```

```
nutrient
Alanine                          Gelatins, dry powder, unsweetened
Arginine                              Seeds, sesame flour, low-fat
Aspartic acid                                  Soy protein isolate
Cystine               Seeds, cottonseed flour, low fat (glandless)
Glutamic acid                                  Soy protein isolate
Glycine                          Gelatins, dry powder, unsweetened
Histidine               Whale, beluga, meat, dried (Alaska Native)
Hydroxyproline    KENTUCKY FRIED CHICKEN, Fried Chicken, ORIGINA...
Isoleucine        Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Leucine           Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Lysine            Seal, bearded (Oogruk), meat, dried (Alaska Na...
Methionine                   Fish, cod, Atlantic, dried and salted
Phenylalanine     Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Proline                          Gelatins, dry powder, unsweetened
Serine            Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Threonine         Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Tryptophan         Sea lion, Steller, meat with fat (Alaska Native)
Tyrosine          Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Valine            Soy protein isolate, PROTEIN TECHNOLOGIES INTE...
Name: food, dtype: object
```

2. 2012년 연방선거관리위원회 데이터베이스

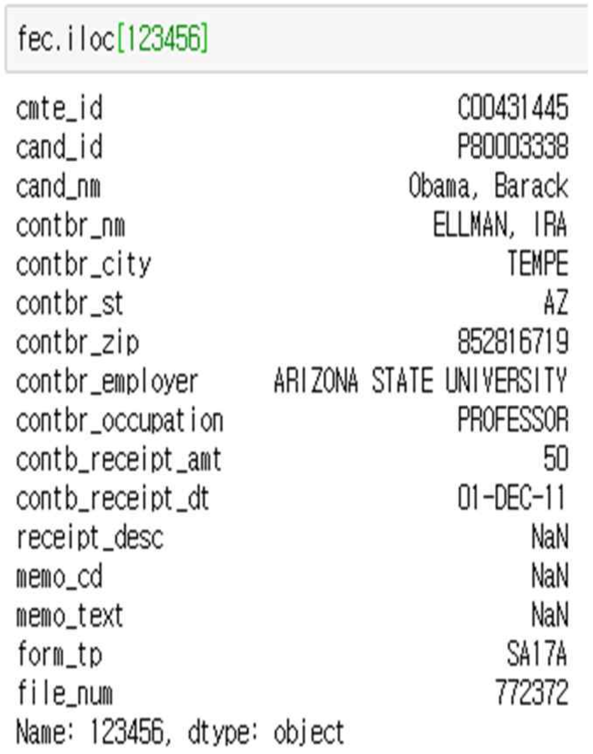
미국연방선거관리위원회는 정치활동 후원금에 대한 데이터를 공개했다.
이 데이터에는 기부자의 이름, 직업, 고용형태, 주소, 기부금액이 포함되어 있다.



```
fec = pd.read_csv('C:/Users/HOME/Desktop/수DA쟁이/Python_for_data_analysis/pydata-book
fec.info()
```

```
C:₩Anaconda3₩lib₩site-packages₩IPython₩core₩interactiveshell.py:3063: DtypeWarning:
Columns (6) have mixed types.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

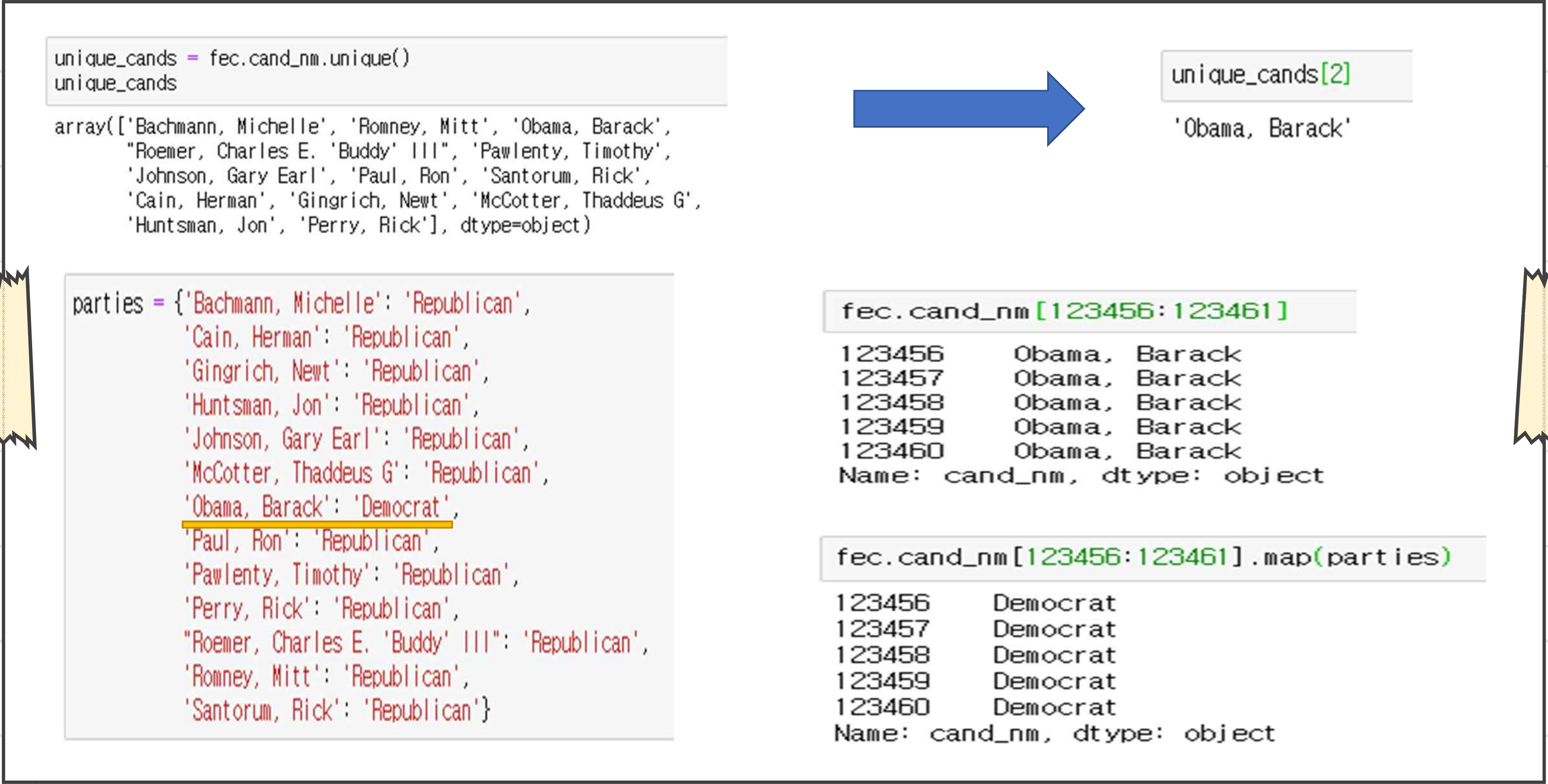
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1001731 entries, 0 to 1001730
Data columns (total 16 columns):
 #   Column             Non-Null Count    Dtype
---  ------             --------------    -----
 0   cmte_id            1001731 non-null  object
 1   cand_id            1001731 non-null  object
 2   cand_nm            1001731 non-null  object
 3   contbr_nm          1001731 non-null  object
 4   contbr_city        1001712 non-null  object
 5   contbr_st          1001727 non-null  object
 6   contbr_zip         1001620 non-null  object
 7   contbr_employer    988002 non-null   object
 8   contbr_occupation  993301 non-null   object
 9   contb_receipt_amt  1001731 non-null  float64
 10  contb_receipt_dt   1001731 non-null  object
 11  receipt_desc       14166 non-null    object
 12  memo_cd            92482 non-null    object
 13  memo_text          97770 non-null    object
 14  form_tp            1001731 non-null  object
 15  file_num           1001731 non-null  int64
dtypes: float64(1), int64(1), object(14)
memory usage: 122.3+ MB
```



```
fec.iloc[123456]
```

```
cmte_id                        C00431445
cand_id                        P80003338
cand_nm                    Obama, Barack
contbr_nm                     ELLMAN, IRA
contbr_city                        TEMPE
contbr_st                             AZ
contbr_zip                     852816719
contbr_employer      ARIZONA STATE UNIVERSITY
contbr_occupation              PROFESSOR
contb_receipt_amt                     50
contb_receipt_dt               01-DEC-11
receipt_desc                         NaN
memo_cd                              NaN
memo_text                            NaN
form_tp                            SA17A
file_num                          772372
Name: 123456, dtype: object
```

2. 2012년 연방선거관리위원회 데이터베이스



```
unique_cands = fec.cand_nm.unique()
unique_cands
```

```
array(['Bachmann, Michelle', 'Romney, Mitt', 'Obama, Barack',
       "Roemer, Charles E. 'Buddy' III", 'Pawlenty, Timothy',
       'Johnson, Gary Earl', 'Paul, Ron', 'Santorum, Rick',
       'Cain, Herman', 'Gingrich, Newt', 'McCotter, Thaddeus G',
       'Huntsman, Jon', 'Perry, Rick'], dtype=object)
```

```
unique_cands[2]
```

```
'Obama, Barack'
```

```
parties = {'Bachmann, Michelle': 'Republican',
           'Cain, Herman': 'Republican',
           'Gingrich, Newt': 'Republican',
           'Huntsman, Jon': 'Republican',
           'Johnson, Gary Earl': 'Republican',
           'McCotter, Thaddeus G': 'Republican',
           'Obama, Barack': 'Democrat',
           'Paul, Ron': 'Republican',
           'Pawlenty, Timothy': 'Republican',
           'Perry, Rick': 'Republican',
           "Roemer, Charles E. 'Buddy' III": 'Republican',
           'Romney, Mitt': 'Republican',
           'Santorum, Rick': 'Republican'}
```

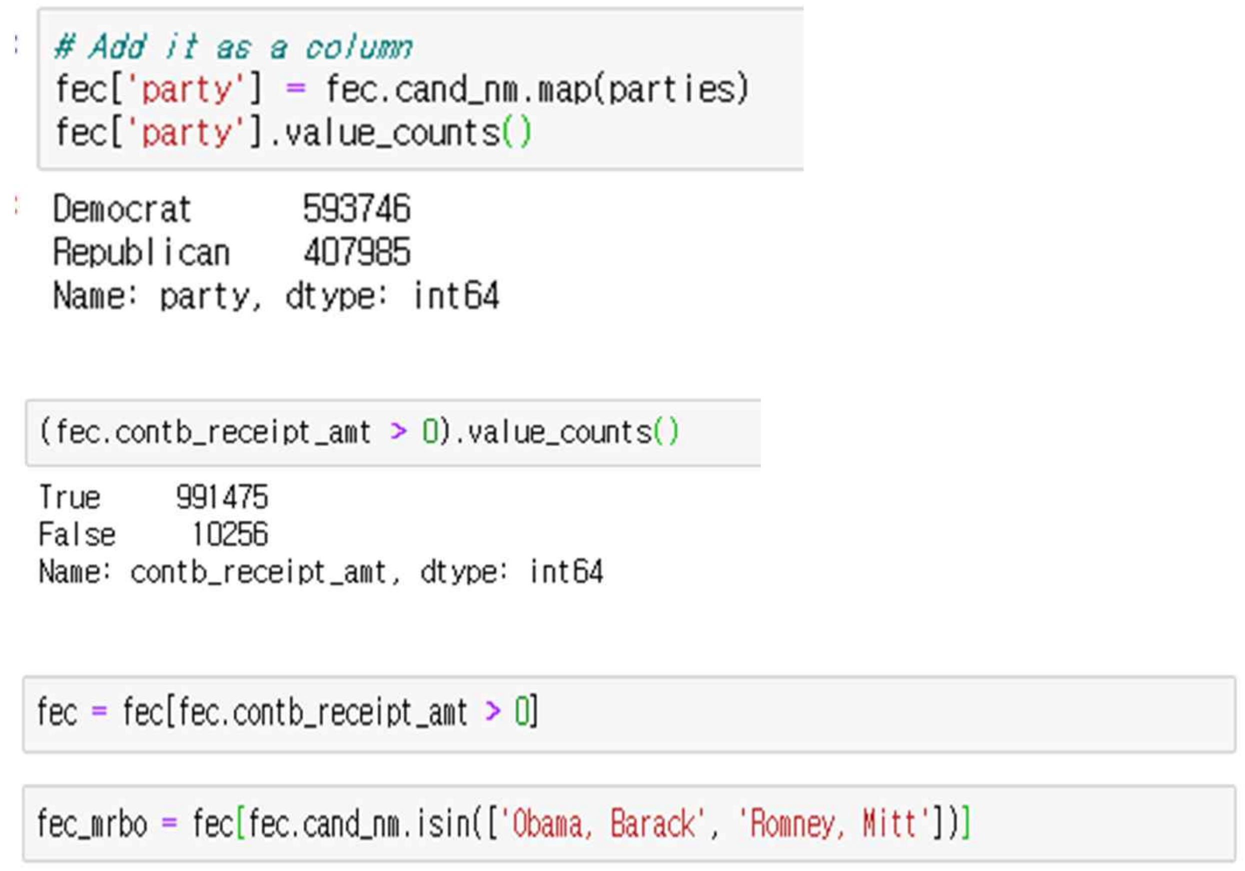
```
fec.cand_nm[123456:123461]
```

```
123456    Obama, Barack
123457    Obama, Barack
123458    Obama, Barack
123459    Obama, Barack
123460    Obama, Barack
Name: cand_nm, dtype: object
```

```
fec.cand_nm[123456:123461].map(parties)
```

```
123456    Democrat
123457    Democrat
123458    Democrat
123459    Democrat
123460    Democrat
Name: cand_nm, dtype: object
```

2. 2012년 연방선거관리위원회 데이터베이스



```
# Add it as a column
fec['party'] = fec.cand_nm.map(parties)
fec['party'].value_counts()
```

```
Democrat      593746
Republican    407985
Name: party, dtype: int64
```

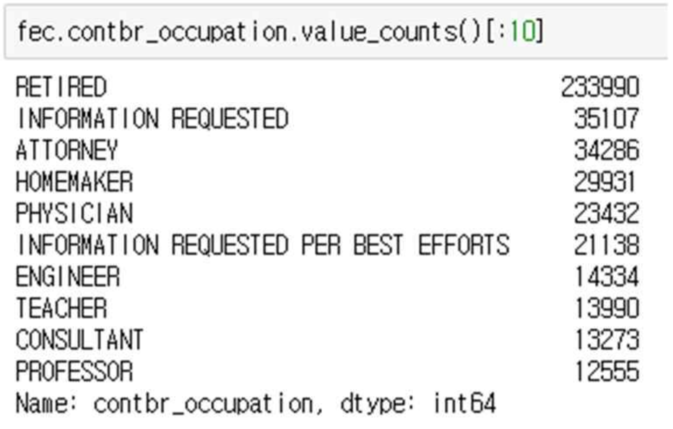
```
(fec.contb_receipt_amt > 0).value_counts()
```

```
True     991475
False     10256
Name: contb_receipt_amt, dtype: int64
```

```
fec = fec[fec.contb_receipt_amt > 0]
```

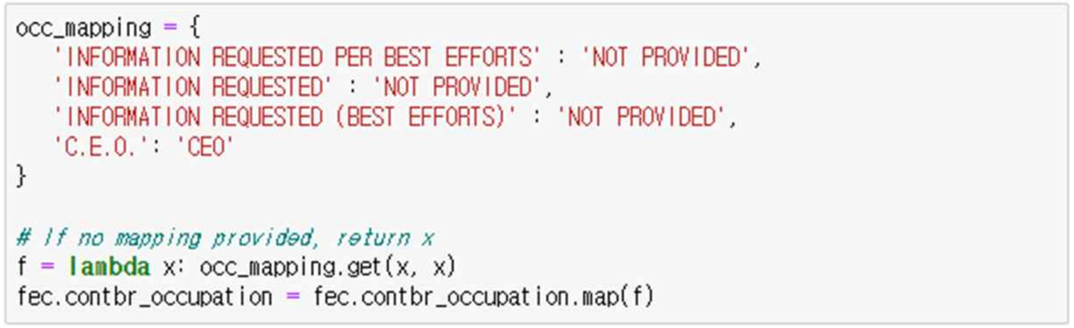
```
fec_mrbo = fec[fec.cand_nm.isin(['Obama, Barack', 'Romney, Mitt'])]
```

3. 직업 및 고용주에 따른 기부 통계

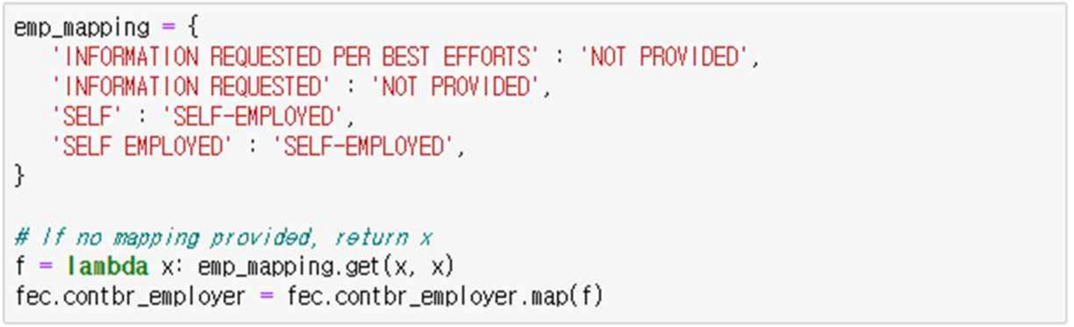


```
fec.contbr_occupation.value_counts()[:10]
```

```
RETIRED                                   233990
INFORMATION REQUESTED                      35107
ATTORNEY                                   34286
HOMEMAKER                                  29931
PHYSICIAN                                  23432
INFORMATION REQUESTED PER BEST EFFORTS     21138
ENGINEER                                   14334
TEACHER                                    13990
CONSULTANT                                 13273
PROFESSOR                                  12555
Name: contbr_occupation, dtype: int64
```

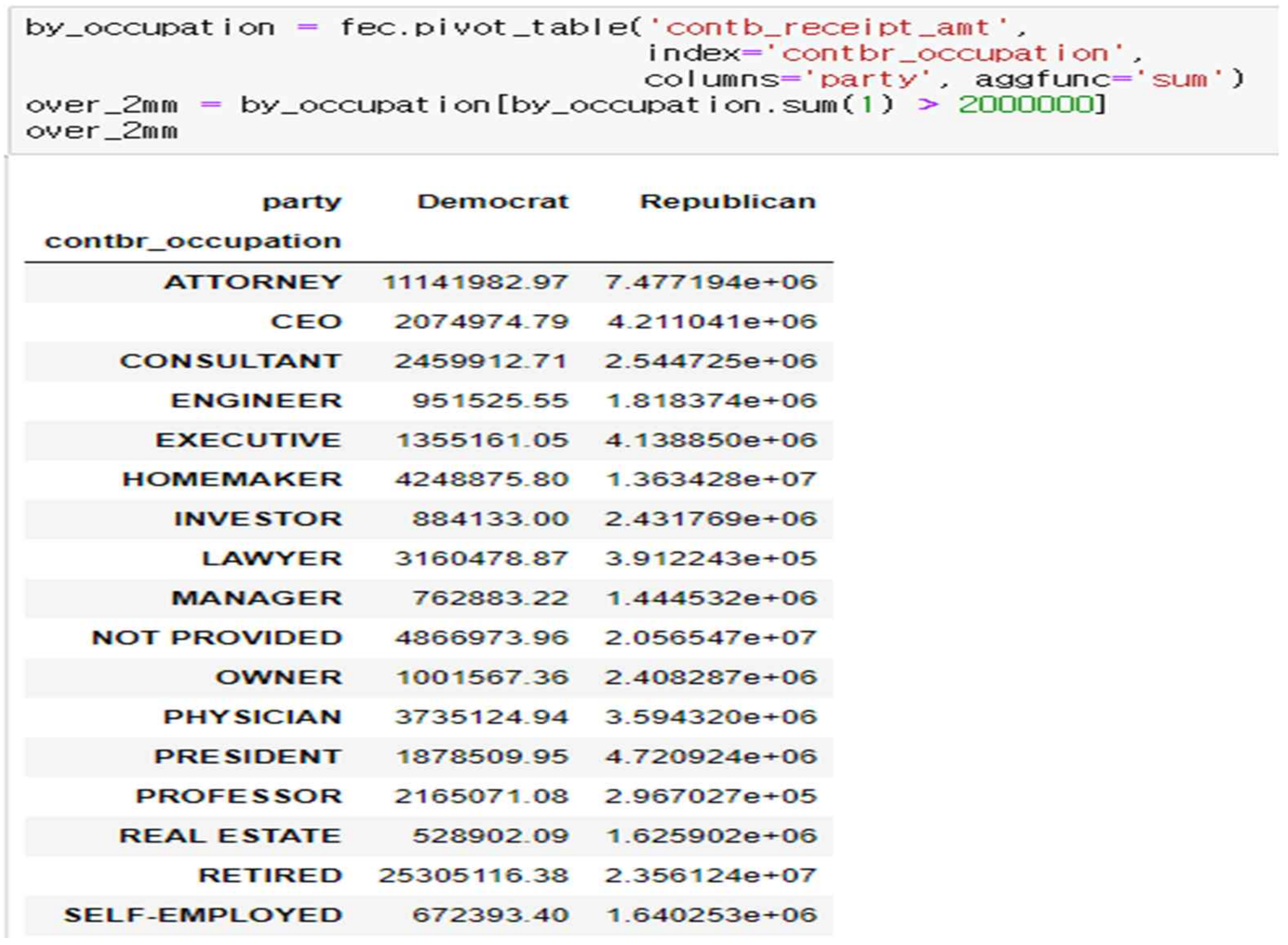


```
occ_mapping = {
    'INFORMATION REQUESTED PER BEST EFFORTS' : 'NOT PROVIDED',
    'INFORMATION REQUESTED' : 'NOT PROVIDED',
    'INFORMATION REQUESTED (BEST EFFORTS)' : 'NOT PROVIDED',
    'C.E.O.': 'CEO'
}

# If no mapping provided, return x
f = lambda x: occ_mapping.get(x, x)
fec.contbr_occupation = fec.contbr_occupation.map(f)
```



```
emp_mapping = {
    'INFORMATION REQUESTED PER BEST EFFORTS' : 'NOT PROVIDED',
    'INFORMATION REQUESTED' : 'NOT PROVIDED',
    'SELF' : 'SELF-EMPLOYED',
    'SELF EMPLOYED' : 'SELF-EMPLOYED',
}

# If no mapping provided, return x
f = lambda x: emp_mapping.get(x, x)
fec.contbr_employer = fec.contbr_employer.map(f)
```

3. 직업 및 고용주에 따른 기부 통계



```python
by_occupation = fec.pivot_table('contb_receipt_amt',
                                index='contbr_occupation',
                                columns='party', aggfunc='sum')
over_2mm = by_occupation[by_occupation.sum(1) > 2000000]
over_2mm
```

party	Democrat	Republican
contbr_occupation		
ATTORNEY	11141982.97	7.477194e+06
CEO	2074974.79	4.211041e+06
CONSULTANT	2459912.71	2.544725e+06
ENGINEER	951525.55	1.818374e+06
EXECUTIVE	1355161.05	4.138850e+06
HOMEMAKER	4248875.80	1.363428e+07
INVESTOR	884133.00	2.431769e+06
LAWYER	3160478.87	3.912243e+05
MANAGER	762883.22	1.444532e+06
NOT PROVIDED	4866973.96	2.056547e+07
OWNER	1001567.36	2.408287e+06
PHYSICIAN	3735124.94	3.594320e+06
PRESIDENT	1878509.95	4.720924e+06
PROFESSOR	2165071.08	2.967027e+05
REAL ESTATE	528902.09	1.625902e+06
RETIRED	25305116.38	2.356124e+07
SELF-EMPLOYED	672393.40	1.640253e+06

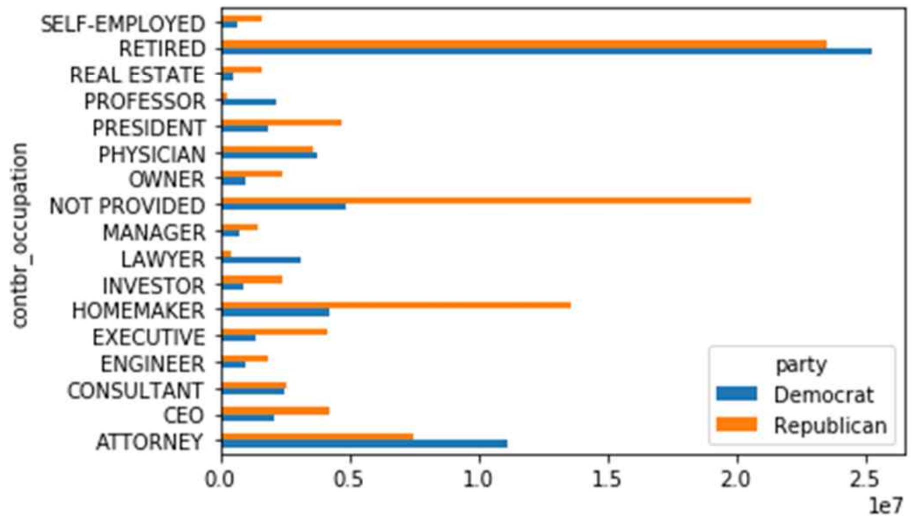
3. 직업 및 고용주에 따른 기부 통계

```
plt.figure()
```

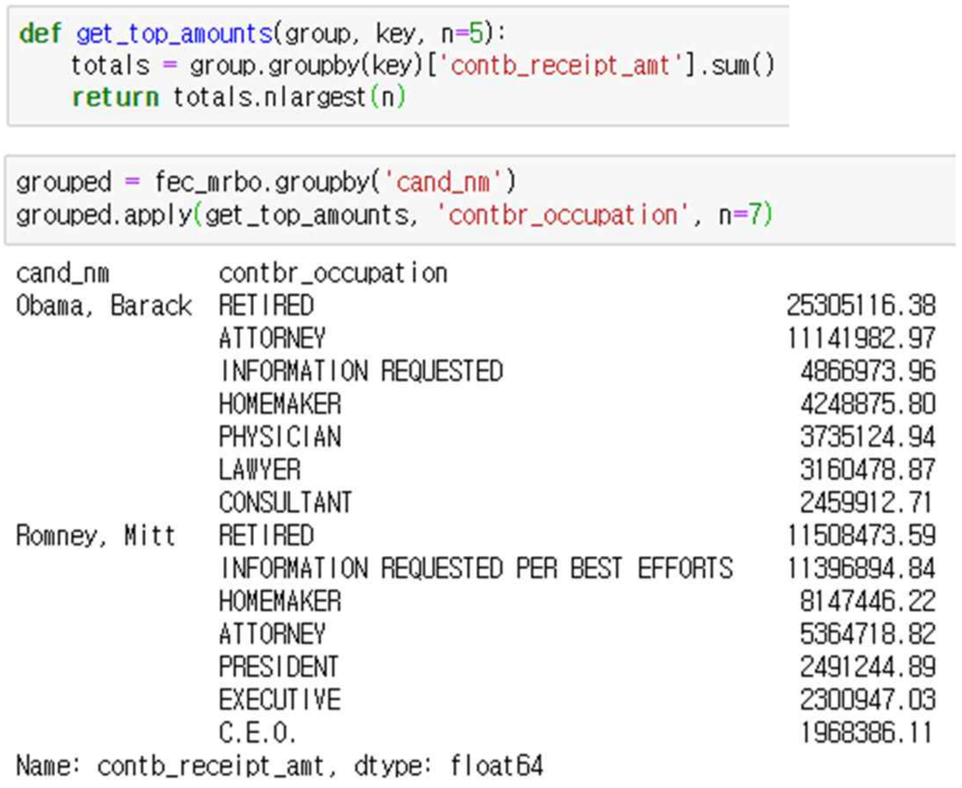
```
<Figure size 432x288 with 0 Axes>

<Figure size 432x288 with 0 Axes>
```

```
over_2mm.plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1810c0f4548>
```



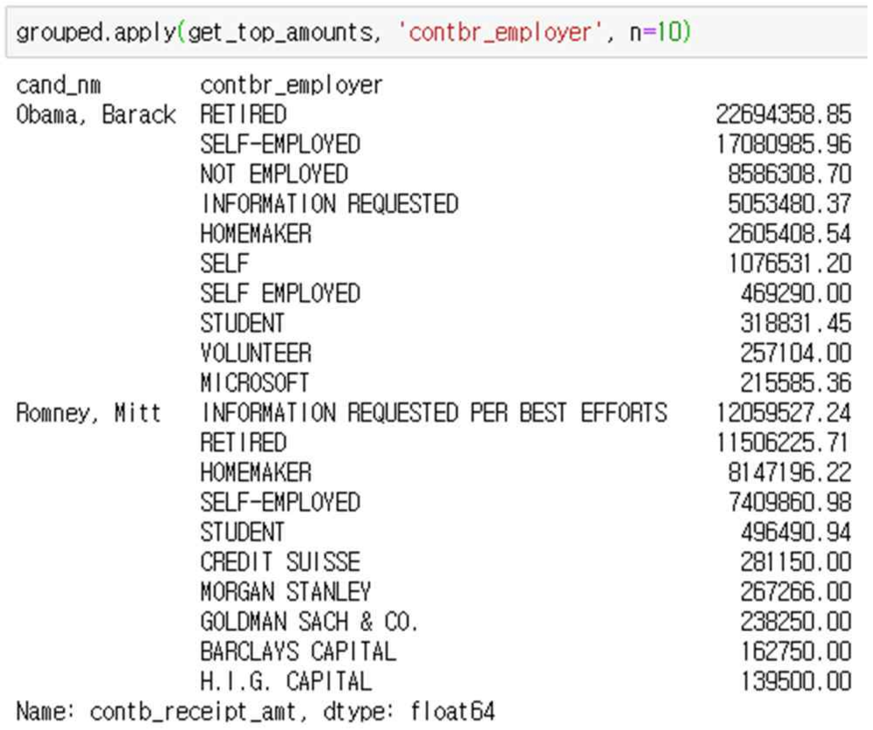
3. 직업 및 고용주에 따른 기부 통계



```
def get_top_amounts(group, key, n=5):
    totals = group.groupby(key)['contb_receipt_amt'].sum()
    return totals.nlargest(n)
```

```
grouped = fec_mrbo.groupby('cand_nm')
grouped.apply(get_top_amounts, 'contbr_occupation', n=7)
```

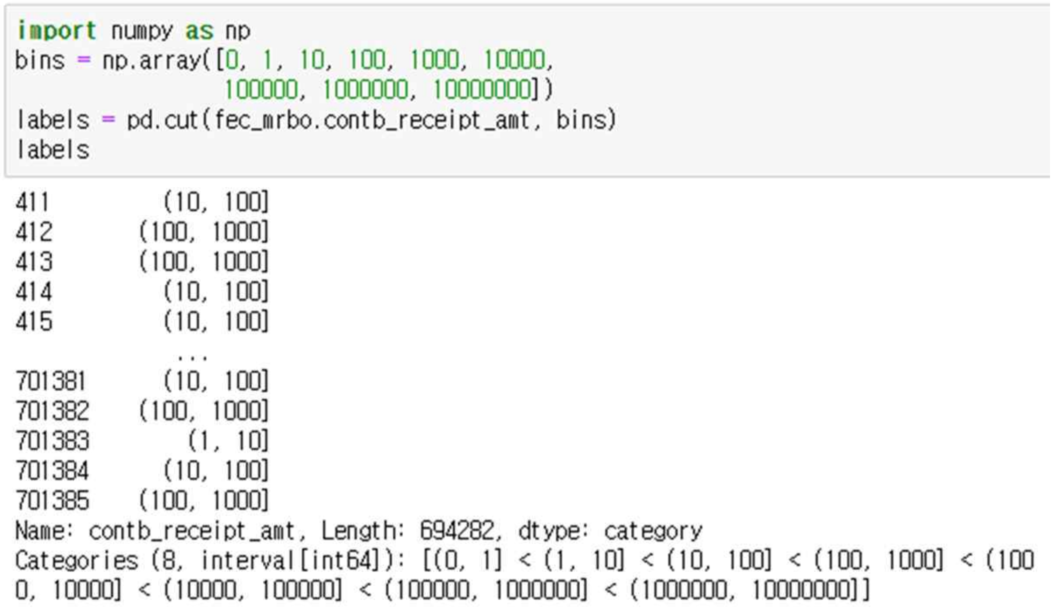
```
cand_nm        contbr_occupation
Obama, Barack  RETIRED                                   25305116.38
               ATTORNEY                                  11141982.97
               INFORMATION REQUESTED                      4866973.96
               HOMEMAKER                                  4248875.80
               PHYSICIAN                                  3735124.94
               LAWYER                                     3160478.87
               CONSULTANT                                 2459912.71
Romney, Mitt   RETIRED                                   11508473.59
               INFORMATION REQUESTED PER BEST EFFORTS    11396894.84
               HOMEMAKER                                  8147446.22
               ATTORNEY                                   5364718.82
               PRESIDENT                                  2491244.89
               EXECUTIVE                                  2300947.03
               C.E.O.                                     1968386.11
Name: contb_receipt_amt, dtype: float64
```



```
grouped.apply(get_top_amounts, 'contbr_employer', n=10)
```

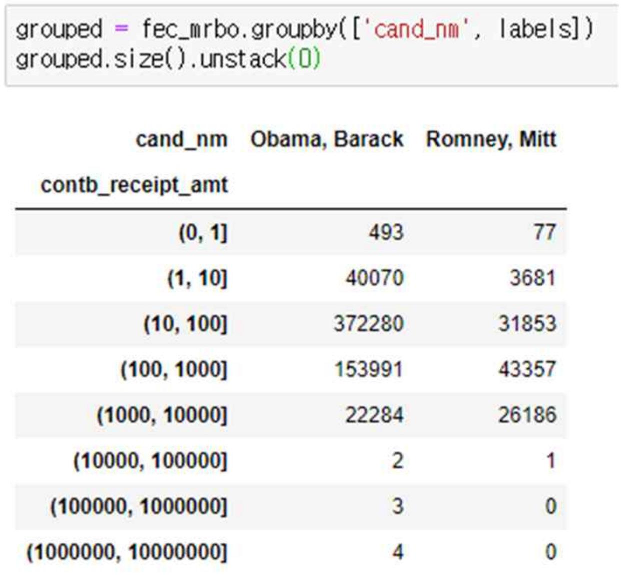
```
cand_nm        contbr_employer
Obama, Barack  RETIRED                                   22694358.85
               SELF-EMPLOYED                             17080985.96
               NOT EMPLOYED                               8586308.70
               INFORMATION REQUESTED                      5053480.37
               HOMEMAKER                                  2605408.54
               SELF                                       1076531.20
               SELF EMPLOYED                               469290.00
               STUDENT                                     318831.45
               VOLUNTEER                                   257104.00
               MICROSOFT                                   215585.36
Romney, Mitt   INFORMATION REQUESTED PER BEST EFFORTS    12059527.24
               RETIRED                                   11506225.71
               HOMEMAKER                                  8147196.22
               SELF-EMPLOYED                              7409860.98
               STUDENT                                     496490.94
               CREDIT SUISSE                               281150.00
               MORGAN STANLEY                              267266.00
               GOLDMAN SACH & CO.                          238250.00
               BARCLAYS CAPITAL                            162750.00
               H.I.G. CAPITAL                              139500.00
Name: contb_receipt_amt, dtype: float64
```

4. 기부 금액



```python
import numpy as np
bins = np.array([0, 1, 10, 100, 1000, 10000,
                 100000, 1000000, 10000000])
labels = pd.cut(fec_mrbo.contb_receipt_amt, bins)
labels
```

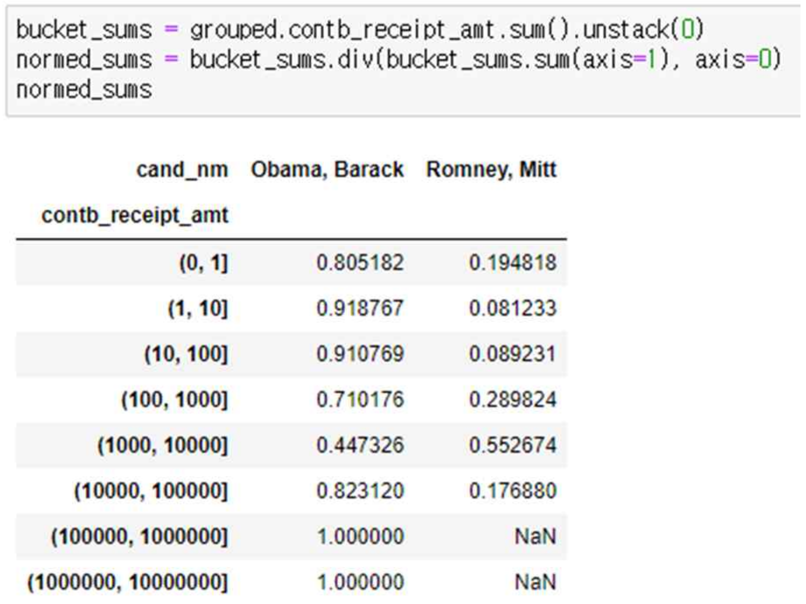
```
411          (10, 100]
412        (100, 1000]
413        (100, 1000]
414          (10, 100]
415          (10, 100]
              ...
701381       (10, 100]
701382     (100, 1000]
701383         (1, 10]
701384       (10, 100]
701385     (100, 1000]
Name: contb_receipt_amt, Length: 694282, dtype: category
Categories (8, interval[int64]): [(0, 1] < (1, 10] < (10, 100] < (100, 1000] < (100
0, 10000] < (10000, 100000] < (100000, 1000000] < (1000000, 10000000]]
```



```python
grouped = fec_mrbo.groupby(['cand_nm', labels])
grouped.size().unstack(0)
```

cand_nm	Obama, Barack	Romney, Mitt
contb_receipt_amt		
(0, 1]	493	77
(1, 10]	40070	3681
(10, 100]	372280	31853
(100, 1000]	153991	43357
(1000, 10000]	22284	26186
(10000, 100000]	2	1
(100000, 1000000]	3	0
(1000000, 10000000]	4	0

4. 기부 금액



```
bucket_sums = grouped.contb_receipt_amt.sum().unstack(0)
normed_sums = bucket_sums.div(bucket_sums.sum(axis=1), axis=0)
normed_sums
```

cand_nm contb_receipt_amt	Obama, Barack	Romney, Mitt
(0, 1]	0.805182	0.194818
(1, 10]	0.918767	0.081233
(10, 100]	0.910769	0.089231
(100, 1000]	0.710176	0.289824
(1000, 10000]	0.447326	0.552674
(10000, 100000]	0.823120	0.176880
(100000, 1000000]	1.000000	NaN
(1000000, 10000000]	1.000000	NaN

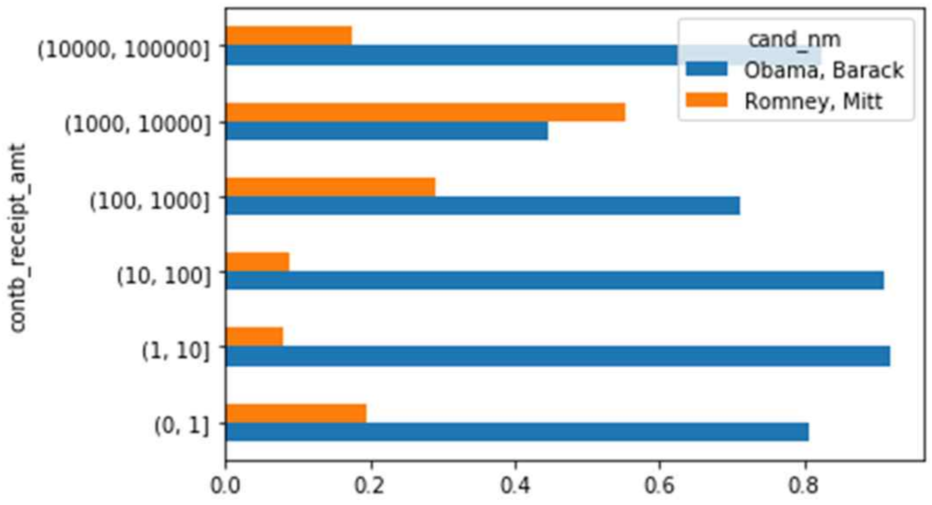
4. 기부 금액

```
plt.figure()
```

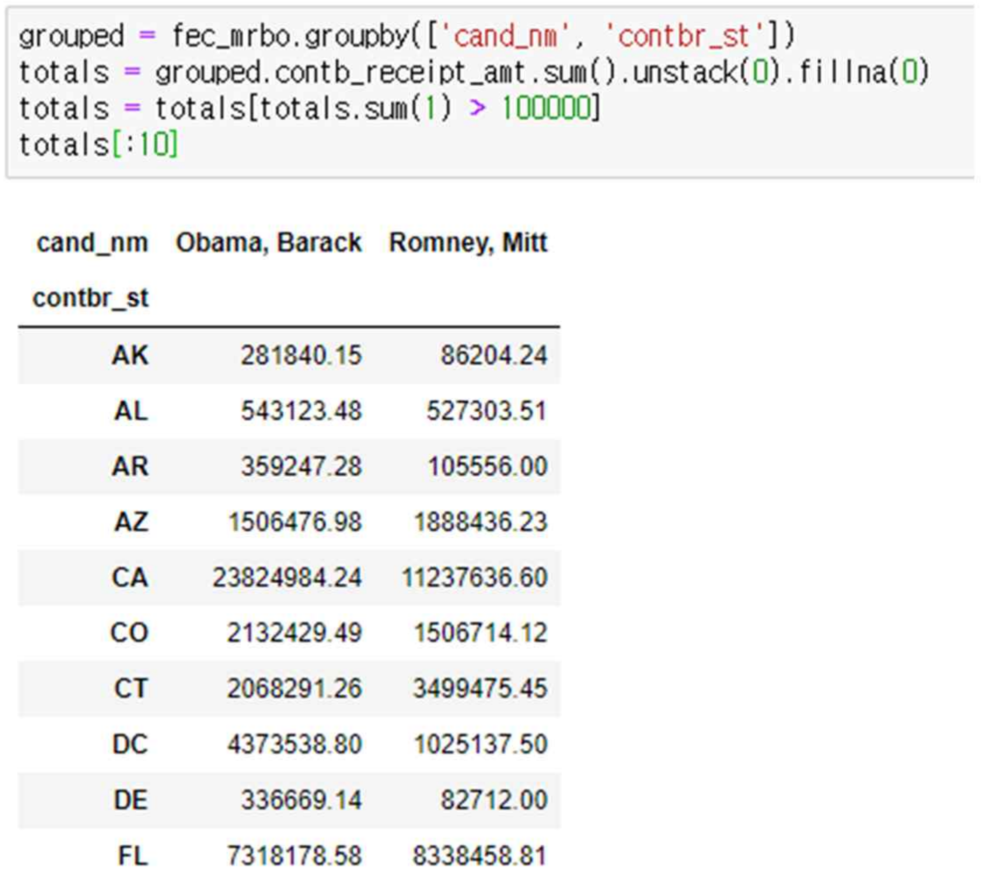
```
<Figure size 432x288 with 0 Axes>

<Figure size 432x288 with 0 Axes>
```

```
normed_sums[:-2].plot(kind='barh')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1810c2c17c8>
```

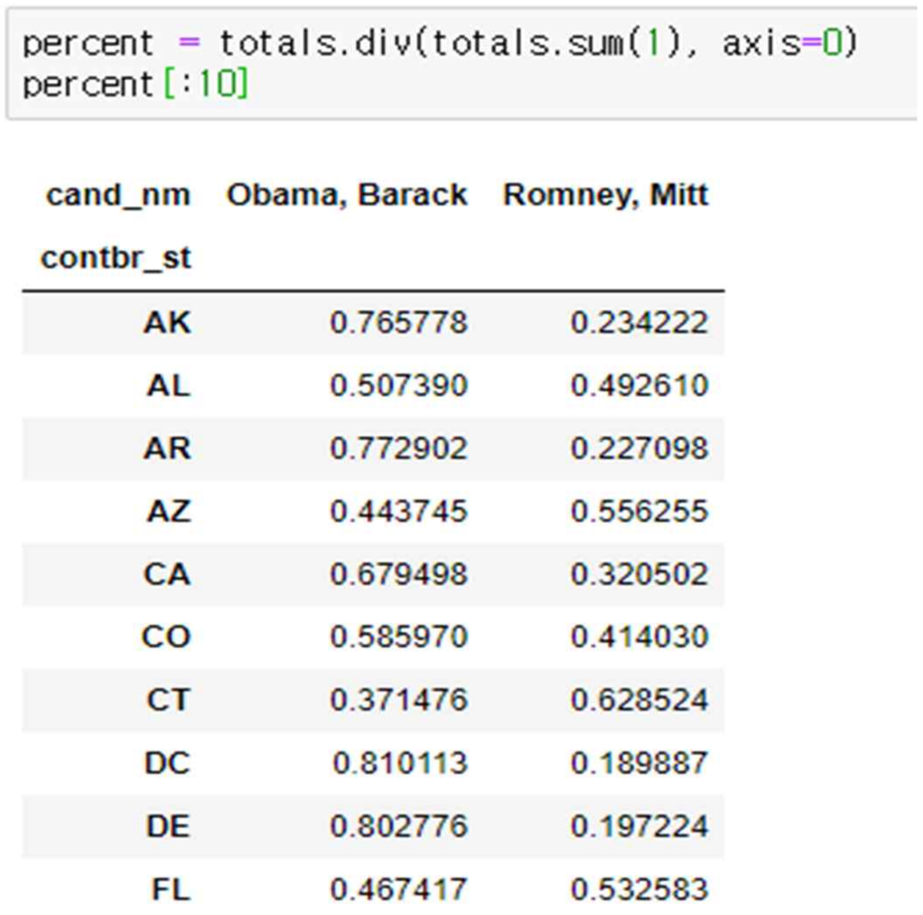


5. 주별 기부 통계



```python
grouped = fec_mrbo.groupby(['cand_nm', 'contbr_st'])
totals = grouped.contb_receipt_amt.sum().unstack(0).fillna(0)
totals = totals[totals.sum(1) > 100000]
totals[:10]
```

```
cand_nm    Obama, Barack  Romney, Mitt
contbr_st
       AK      281840.15      86204.24
       AL      543123.48     527303.51
       AR      359247.28     105556.00
       AZ     1506476.98    1888436.23
       CA    23824984.24   11237636.60
       CO     2132429.49    1506714.12
       CT     2068291.26    3499475.45
       DC     4373538.80    1025137.50
       DE      336669.14      82712.00
       FL     7318178.58    8338458.81
```



```python
percent = totals.div(totals.sum(1), axis=0)
percent[:10]
```

```
cand_nm    Obama, Barack  Romney, Mitt
contbr_st
       AK       0.765778      0.234222
       AL       0.507390      0.492610
       AR       0.772902      0.227098
       AZ       0.443745      0.556255
       CA       0.679498      0.320502
       CO       0.585970      0.414030
       CT       0.371476      0.628524
       DC       0.810113      0.189887
       DE       0.802776      0.197224
       FL       0.467417      0.532583
```

END!