TheDataLytics

python™

ANUURAG EDLABADKAR

pandas

#**PRMCEM - Prof Ram Meghe College of Engineering & Management, Badnera - Amravati**

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

#Project: 1 **Occupation**

```
users = 
pd.read_table('https://raw.githubusercontent.com/justmarkham/DAT8/
master/data/u.user', 
                      sep='|', index_col='user_id')
```

```
users
```

```
         age gender     occupation zip_code
user_id
1         24      M     technician    85711
2         53      F          other    94043
3         23      M         writer    32067
4         24      M     technician    43537
5         33      F          other    15213
...      ...    ...            ...      ...
939       26      F        student    33319
940       32      M  administrator    02215
941       20      M        student    97229
942       48      F      librarian    78209
943       22      M        student    77841

[943 rows x 4 columns]
```

```
users.head(25)
```

```
         age gender     occupation zip_code
user_id
1         24      M     technician    85711
2         53      F          other    94043
3         23      M         writer    32067
4         24      M     technician    43537
5         33      F          other    15213
6         42      M      executive    98101
7         57      M  administrator    91344
8         36      M  administrator    05201
9         29      M        student    01002
10        53      M         lawyer    90703
11        39      F          other    30329
12        28      F          other    06405
13        47      M       educator    29206
14        45      M      scientist    55106
15        49      F       educator    97301
16        21      M  entertainment    10309
17        30      M     programmer    06355
18        35      F          other    37212
19        40      M      librarian    02138
20        42      F      homemaker    95660
21        26      M         writer    30068
22        25      M         writer    40206
23        30      F         artist    48197
24        21      F         artist    94533
25        39      M       engineer    55107
```

```
users.tail(25)
```

```
         age gender     occupation zip_code
user_id
919       25      M          other    14216
```

```
920       30      F         artist    90008
921       20      F        student    98801
922       29      F  administrator    21114
923       21      M        student    E2E3R
924       29      M          other    11753
925       18      F       salesman    49036
926       49      M  entertainment    01701
927       23      M     programmer    55428
928       21      M        student    55408
929       44      M      scientist    53711
930       28      F      scientist    07310
931       60      M       educator    33556
932       58      M       educator    06437
933       28      M        student    48105
934       61      M       engineer    22902
935       42      M         doctor    66221
936       24      M          other    32789
937       48      M       educator    98072
938       38      F     technician    55038
939       26      F        student    33319
940       32      M  administrator    02215
941       20      M        student    97229
942       48      F      librarian    78209
943       22      M        student    77841
```

```
users.shape
```

```
(943, 4)
```

```
users.shape[0]
```

```
943
```

```
users.shape[1]
```

```
4
```

```
users.info
```

```
<bound method DataFrame.info of          age gender     occupation
zip_code
user_id
1         24      M     technician    85711
2         53      F          other    94043
3         23      M         writer    32067
4         24      M     technician    43537
5         33      F          other    15213
...      ...    ...            ...      ...
939       26      F        student    33319
940       32      M  administrator    02215
941       20      M        student    97229
942       48      F      librarian    78209
```

```
943       22      M        student    77841

[943 rows x 4 columns]>
```

```
users.columns
```

```
Index(['age', 'gender', 'occupation', 'zip_code'], dtype='object')
```

```
users.describe
```

```
<bound method NDFrame.describe of          age gender     occupation
zip_code
user_id
1         24      M     technician    85711
2         53      F          other    94043
3         23      M         writer    32067
4         24      M     technician    43537
5         33      F          other    15213
...      ...    ...            ...      ...
939       26      F        student    33319
940       32      M  administrator    02215
941       20      M        student    97229
942       48      F      librarian    78209
943       22      M        student    77841

[943 rows x 4 columns]>
```

```
users.describe()
```

```
              age
count  943.000000
mean    34.051962
std     12.192740
min      7.000000
25%     25.000000
50%     31.000000
75%     43.000000
max     73.000000
```

```
users.index
```

```
Int64Index([  1,   2,   3,   4,   5,   6,   7,   8,   9,  10,
            ...
            934, 935, 936, 937, 938, 939, 940, 941, 942, 943],
           dtype='int64', name='user_id', length=943)
```

```
users.dtypes
```

```
age            int64
gender        object
occupation    object
```

```
zip_code      object
dtype: object
```

```
users.occupation
```

```
users['occupation']
```

```
user_id
1         technician
2              other
3             writer
4         technician
5              other
           ...      
939          student
940    administrator
941          student
942        librarian
943          student
Name: occupation, Length: 943, dtype: object
```

```
users[['age','occupation','zip_code','gender']]
```

```
         age     occupation zip_code gender
user_id                                    
1         24     technician    85711      M
2         53          other    94043      F
3         23         writer    32067      M
4         24     technician    43537      M
5         33          other    15213      F
...      ...            ...      ...    ...
939       26        student    33319      F
940       32  administrator    02215      M
941       20        student    97229      M
942       48      librarian    78209      F
943       22        student    77841      M

[943 rows x 4 columns]
```

```
users.occupation.unique()
```

```
array(['technician', 'other', 'writer', 'executive', 'administrator',
       'student', 'lawyer', 'educator', 'scientist', 'entertainment',
       'programmer', 'librarian', 'homemaker', 'artist', 'engineer',
       'marketing', 'none', 'healthcare', 'retired', 'salesman',
'doctor'],
      dtype=object)
```

```
users.gender.unique()
```

```
array(['M', 'F'], dtype=object)
```

```
users.age.unique()
```

```
array([24, 53, 23, 33, 42, 57, 36, 29, 39, 28, 47, 45, 49, 21, 30, 35,
40,
       26, 25, 32, 41,  7, 38, 20, 19, 27, 18, 22, 37, 16, 50, 31, 51,
17,
       48, 34, 43, 60, 55, 15, 61, 44, 54, 59, 46, 13, 52, 56, 14, 66,
62,
       11, 65, 68, 63, 64, 10, 73, 58, 69, 70])
```

```
users.zip_code.unique()
```

```
array(['85711', '94043', '32067', '43537', '15213', '98101', '91344',
       '05201', '01002', '90703', '30329', '06405', '29206', '55106',
       '97301', '10309', '06355', '37212', '02138', '95660', '30068',
       '40206', '48197', '94533', '55107', '21044', '30030', '55369',
       '55436', '10003', '78741', '27510', '42141', '42459', '93117',
       '55105', '54467', '01040', '27514', '80525', '17870', '20854',
       '46260', '50233', '46538', '07102', '12550', '76111', '52245',
       '16509', '55414', '66315', '01331', '84010', '52246', '08403',
       '06472', '30040', '97214', '75240', '43202', '48118', '80521',
       '60402', '22904', '55337', '60067', '98034', '73034', '41850',
       'T8H1N', '08816', '02215', '29379', '61801', '03755', '52241',
       '21218', '22902', '44133', '20003', '46005', '89503', '11701',
       '68106', '78155', '01913', '23112', '71457', '10707', '75206',
       '98006', '90291', '63129', '90254', '05146', '30220', '55108',
       '55125', '60466', '63130', '55423', '77840', '90630', '60613',
       '95032', '75013', '17110', '97232', '16125', '90210', '67401',
       '06260', '99603', '22206', '20008', '60615', '22202', '20015',
       '73439', '20009', '07039', '60115', '15237', '94612', '78602',
       '80236', '38401', '97365', '84408', '53211', '08904', '32250',
       '36117', '08832', '20910', 'V3N4P', '83814', '02143', '97006',
       '17325', '02139', '48103', '68767', '60641', '53703', '11217',
       '08360', '70808', '27606', '55346', '66215', '55104', '15610',
       '97212', '80123', '53715', '55113', 'L9G2B', '80127', '53705',
       '30067', '78750', '22207', '22306', '52302', '21911', '07030',
       '19104', '49512', '20755', '60202', '33884', '27708', '76013',
       '97403', '00000', '16801', '29440', '95014', '95938', '95161',
       '90840', '49931', '02154', '93555', '75094', '17604', '93402',
       'E2A4H', '60201', '32301', '10960', '06371', '53115', '92037',
       '01720', '85710', '03060', '32605', '61401', '55345', '11231',
       '63033', '11727', '06513', '43212', '78205', '20685', '27502',
       '47906', '43512', '58202', '92103', '60659', '22003', '22903',
       '14476', '01080', '99709', '98682', '94702', '22973', '53214',
       '63146', '44124', '95628', '20784', '20001', '31404', '55109',
       '28734', '20770', '37235', '84103', '95110', '85032', '07733',
       '42647', '07029', '39042', '77005', '77801', '48823', '89801',
       '85202', '78264', '90064', '84601', '78756', '83716', '19422',
       '43201', '63119', '22932', '53706', '10016', '92064', '95064',
       '55406', '30033', '85251', '06059', '20057', '55305', '92629',
       '53713', '15217', '31211', '23226', '94619', '93550', '44106',
       '94703', '60804', '92110', '50325', '16803', '98103', '01581',
       '63108', '55439', '77904', '14853', '71701', '94086', '73132',
       '55454', '95076', '70802', '91711', '73071', '02110', '60035',
       '08043', '18301', '77009', '13210', '06518', '22030', '24060',
       '55413', '50613', '19149', '02176', '15235', '11101', '06779',
       '40504', 'V0R2M', '30002', '33775', '42101', '10522', '59717',
       '37901', '44405', '30093', '94117', '94143', '76059', '45660',
       '61455', '49938', '28480', '60135', '92688', '98133', '10022',
       '98027', '44074', '85233', '87501', '01810', '50670', '37411',
       '92113', '91335', '08534', '99206', '66046', '55116', '78746',
       '37777', '10010', '18015', '02859', '98117', '55117', '94608',
       '01824', '75204', '45218', '43221', '37412', '36106', '83702',
       '85016', '84604', '59801', '83686', '96819', '44092', '94551',
       '60008', '92374', '78213', '84107', '95129', '06811', '10019',
       '93109', '03261', '61755', '98225', '94025', '44691', '15222',
       '78212', '38115', '92626', '21206', '43215', '02140', '91606',
       '55422', '58644', '01602', '85258', '29205', '98199', '50311',
       '11211', '49705', '60007', '17345', '43204', '20817', '48076',
       '55013', '85282', '33308', '53202', '92653', '10021', '55021',
       '11758', '48446', '28018', '06333', '97330', '83709', '31820',
       '30011', 'Y1A6B', '29201', '60630', '98102', '02918', '75218',
       '94583', '05001', '90804', '91201', '02341', '78628', '77459',
       '87544', '94708', '93711', '75230', '60440', '02125', '55409',
       '98257', '37771', '40256', '21208', '95821', '93101', '92121',
       '21012', 'V5A2B', '53711', '94618', '60090', '49428', '03052',
       '50112', '55408', '75006', '94305', '10025', '23092', '92115',
       '20657', '03869', '28450', '19382', '10011', '98038', '21250',
       '20090', '26241', '20707', '49508', '55320', '12603', '02146',
       '55443', '04102', '02159', '19711', '97124', '12180', '44224',
       '94040', '97408', '92705', '02324', '05464', '80302', '30078',
       '21010', '80303', '84302', '60515', '95123', '29464', '08052',
       '22911', '14534', '95468', '45680', '95453', '68147', '62901',
       '23227', '30606', '63132', '60005', '20879', '32707', '94591',
       '14627', '01915', '91903', '01945', '48911', '53188', '46032',
       '98281', '77845', 'M7A1A', '17961', '94131', '93003', '29631',
       '27511', '98501', '79508', '14216', '93063', '90034', '82435',
       '92093', '97520', 'M4J2K', '31909', '77073', '84116', '43085',
       'R3T5K', '02320', '99687', '34656', '47905', '11787', '33716',
       '63044', '21227', '77008', '79070', '29678', '80227', '27705',
       '11201', '44212', '44134', '81648', '14850', '60187', '20723',
       '19807', '08034', '94306', '38866', '23237', '48043', '74101',
       '01940', '12065', '60626', '95521', '55122', '63645', '51250',
       '45810', '91351', '39762', '02903', '78739', '60657', '10314',
       '78704', '54248', '77380', '98121', '19102', '19341', '94115',
       '55412', '61820', '01970', '21114', '91919', '90095', '22906',
       '28814', '32712', '99835', '61462', '54302', '90405', '97208',
       '55128', '23509', '26506', '27713', '60476', '45439', '63304',
       '60089', '18053', '85210', '06365', '94920', '77042', '06906',
       '96754', '76309', '56321', '89104', '91105', '54494', '19146',
       '96349', 'N4T1A', '92020', '15203', '54901', '07204', '55343',
       '91206', '44265', '84105', '64118', 'V0R2H', '16506', '11238',
       '17331', '94403', '40243', '80538', '56567', '32114', '70403',
       '98405', '85719', '98072', '95403', '73162', '29210', '92660',
       '47024', '19047', '93612', '94720', '80919', '32303', '21201',
       '97007', '90247', '68503', '14211', '97302', '95050', '02113',
       '62903', '33066', '12866', '06927', '15232', '27105', '80027',
       '90036', '51157', '01960', 'K7L5J', '94560', '48825', '33205',
       '77081', '91040', '23322', '01754', '98620', '05779', '55420',
       '80913', '20064', '12205', '85281', '57197', '08610', '33755',
       '62522', '64131', '19716', '92154', '34105', '90016', '30803',
       '80526', '73013', '76234', '02136', '12345', '28806', '60152',
       '40205', '37725', '53144', '50322', '15017', '05452', '77048',
       '80228', '80209', '53066', '33765', '90019', '64153', '11577',
       '10018', '01375', '90814', '47401', '93055', '95662', '97405',
       '47130', '55417', '25652', '78390', '29646', '40515', '04988',
       '97215', 'V1G4L', '09645', '06492', '48322', '14085', '13820',
       '63021', '60302', '92507', '55303', '65203', '44648', '74078',
       '33763', '37076', '35802', '20902', '77504', '43017', '40503',
       '50266', '95316', '27249', '17036', '03062', '45243', '95823',
       '74075', '91505', '33484', '18505', 'L1V3W', '97203', '20850',
       '61073', '30350', '70124', '68504', '53171', '29301', '53210',
       '06512', '76201', '08105', '60614', 'N2L5N', '20006', '70116',
       '90008', '98801', 'E2E3R', '11753', '49036', '01701', '55428',
       '07310', '33556', '06437', '48105', '66221', '32789', '55038',
       '33319', '97229', '78209', '77841'], dtype=object)
```

```
len(users.occupation.unique())
```

```
21
```

```
users.occupation.value_counts().head()
```

```
student          196
other            105
educator          95
administrator     79
engineer          67
Name: occupation, dtype: int64
```

```
users.occupation.value_counts().head(25)
```

```
student          196
other            105
educator          95
administrator     79
engineer          67
programmer        66
librarian         51
writer            45
executive         32
scientist         31
```

```
artist            28
technician        27
marketing         26
entertainment     18
healthcare        16
retired           14
lawyer            12
salesman          12
none               9
homemaker          7
doctor             7
Name: occupation, dtype: int64
```

```
users.describe(include = "all")
```

```
               age gender occupation zip_code
count   943.000000    943        943      943
unique         NaN      2         21      795
top            NaN      M    student    55414
freq           NaN    670        196        9
mean     34.051962    NaN        NaN      NaN
std      12.192740    NaN        NaN      NaN
min       7.000000    NaN        NaN      NaN
25%      25.000000    NaN        NaN      NaN
50%      31.000000    NaN        NaN      NaN
75%      43.000000    NaN        NaN      NaN
max      73.000000    NaN        NaN      NaN
```

```
users.occupation.describe()
```

```
count         943
unique         21
top       student
freq          196
Name: occupation, dtype: object
```

```
users.gender.describe()
```

```
count     943
unique      2
top         M
freq      670
Name: gender, dtype: object
```

```
users.zip_code.describe()
```

```
count       943
unique      795
top       55414
freq          9
Name: zip_code, dtype: object
```

```
users.age.describe()
```

```
count    943.000000
mean      34.051962
std       12.192740
min        7.000000
25%       25.000000
50%       31.000000
75%       43.000000
max       73.000000
Name: age, dtype: float64
```

#**Project 2: Chipotle Dataset**

```
url = 'https://raw.githubusercontent.com/justmarkham/DAT8/master/data/chipotle.tsv'
```

```
chipo = pd.read_csv(url, sep = '\t')
```

```
chipo
```

```
      order_id  ...  item_price
0            1  ...      $2.39
1            1  ...      $3.39
2            1  ...      $3.39
3            1  ...      $2.39
4            2  ...     $16.98
...        ...  ...         ...
4617      1833  ...     $11.75
4618      1833  ...     $11.75
4619      1834  ...     $11.25
4620      1834  ...      $8.75
4621      1834  ...      $8.75

[4622 rows x 5 columns]
```

```
chipo.head(), chipo.tail()
```

```
(   order_id  ...  item_price
 0         1  ...      $2.39
 1         1  ...      $3.39
 2         1  ...      $3.39
 3         1  ...      $2.39
 4         2  ...     $16.98

 [5 rows x 5 columns],       order_id  ...  item_price
 4617      1833  ...     $11.75
 4618      1833  ...     $11.75
 4619      1834  ...     $11.25
 4620      1834  ...      $8.75
 4621      1834  ...      $8.75
 [5 rows x 5 columns])
```

```
chipo.info, chipo.describe, chipo.shape, chipo.describe()
```

```
(<bound method DataFrame.info of       order_id  ...  item_price
 0           1  ...      $2.39
 1           1  ...      $3.39
 2           1  ...      $3.39
 3           1  ...      $2.39
 4           2  ...     $16.98
 ...       ...  ...         ...
 4617     1833  ...     $11.75
 4618     1833  ...     $11.75
 4619     1834  ...     $11.25
 4620     1834  ...      $8.75
 4621     1834  ...      $8.75

 [4622 rows x 5 columns]>,
 <bound method NDFrame.describe of       order_id  ...  item_price
 0           1  ...      $2.39
 1           1  ...      $3.39
 2           1  ...      $3.39
 3           1  ...      $2.39
 4           2  ...     $16.98
 ...       ...  ...         ...
 4617     1833  ...     $11.75
 4618     1833  ...     $11.75
 4619     1834  ...     $11.25
 4620     1834  ...      $8.75
 4621     1834  ...      $8.75

 [4622 rows x 5 columns]>,
 (4622, 5),
           order_id     quantity
 count  4622.000000  4622.000000
 mean    927.254868     1.075725
 std     528.890796     0.410186
 min       1.000000     1.000000
 25%     477.250000     1.000000
 50%     926.000000     1.000000
 75%    1393.000000     1.000000
 max    1834.000000    15.000000)
```

```
chipo.dtypes
```

```
order_id               int64
quantity               int64
item_name             object
choice_description    object
```

```
item_price            object
dtype: object
```

```
chipo.columns
```

```
Index(['order_id', 'quantity', 'item_name', 'choice_description',
       'item_price'],
      dtype='object')
```

```
chipo.order_id, chipo.quantity, chipo.item_name,
chipo.choice_description, chipo.item_price
```

```
(0          1
 1          1
 2          1
 3          1
 4          2
         ...
 4617    1833
 4618    1833
 4619    1834
 4620    1834
 4621    1834
 Name: order_id, Length: 4622, dtype: int64, 0       1
 1       1
 2       1
 3       1
 4       2
        ..
 4617    1
 4618    1
 4619    1
 4620    1
 4621    1
 Name: quantity, Length: 4622, dtype: int64, 0                Chips
and Fresh Tomato Salsa
 1                                        Izze
 2                            Nantucket Nectar
 3       Chips and Tomatillo-Green Chili Salsa
 4                                Chicken Bowl
                         ...
 4617                            Steak Burrito
 4618                            Steak Burrito
 4619                       Chicken Salad Bowl
 4620                       Chicken Salad Bowl
 4621                       Chicken Salad Bowl
 Name: item_name, Length: 4622, dtype: object, 0
NaN
 1                                            [Clementine]
 2                                                 [Apple]
 3                                                     NaN
 4       [Tomatillo-Red Chili Salsa (Hot), [Black Beans...
                               ...
 4617    [Fresh Tomato Salsa, [Rice, Black Beans, Sour ...
 4618    [Fresh Tomato Salsa, [Rice, Sour Cream, Cheese...
 4619    [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...
 4620    [Fresh Tomato Salsa, [Fajita Vegetables, Lettu...
 4621    [Fresh Tomato Salsa, [Fajita Vegetables, Pinto...
 Name: choice_description, Length: 4622, dtype: object, 0        $2.39

 1        $3.39
 2        $3.39
 3        $2.39
 4       $16.98
          ...
 4617    $11.75
 4618    $11.75
 4619    $11.25
 4620     $8.75
 4621     $8.75
 Name: item_price, Length: 4622, dtype: object)
```

```
chipo[['order_id', 'quantity', 'item_name',
'choice_description','item_price']]
```

```
      order_id  ...  item_price
0            1  ...       $2.39
1            1  ...       $3.39
2            1  ...       $3.39
3            1  ...       $2.39
4            2  ...      $16.98
...        ...  ...         ...
4617      1833  ...      $11.75
4618      1833  ...      $11.75
4619      1834  ...      $11.25
4620      1834  ...       $8.75
4621      1834  ...       $8.75

[4622 rows x 5 columns]
```

```
mostOrd = chipo.item_name.value_counts().max()
mostOrd
```

```
726
```

```
chipo.choice_description.value_counts().head()
```

```
[Diet Coke]
134
[Coke]
123
[Sprite]
```

```
77
[Fresh Tomato Salsa, [Rice, Black Beans, Cheese, Sour Cream, Lettuce]]
42
[Fresh Tomato Salsa, [Rice, Black Beans, Cheese, Sour Cream,
Guacamole, Lettuce]]     40
Name: choice_description, dtype: int64
```

```
chipo.choice_description.value_counts()
```

```
chipo.item_name.value_counts()
```

```
dollarizer = lambda x: float(x[1:-1])
chipo.item_price = chipo.item_price.apply(dollarizer)
```

```
chipo.dtypes
```

```
chipo.item_price.sum()
```

```
34500.16
```

```
chipo.order_id.value_counts().count()
```

```
1834
```

```
order_grouped = chipo.groupby(by=['order_id']).sum()
order_grouped.mean()['item_price']
```

```
18.81142857142869
```

```
chipo.item_name.value_counts().count()
```

```
50
```

```
chipo.item_name.value_counts()
```

Filter and Sort

```
chipo10 = chipo[chipo['item_price'] > 10.00]
chipo10
```

```
      order_id  ...  item_price
4            2  ...       16.98
5            3  ...       10.98
7            4  ...       11.75
13           7  ...       11.25
23          12  ...       10.98
...        ...  ...         ...
4610      1830  ...       11.75
4611      1830  ...       11.25
4617      1833  ...       11.75
4618      1833  ...       11.75
4619      1834  ...       11.25
```

```
[1130 rows x 5 columns]
```

```
chipo10 = chipo[chipo['item_name'] == 'Chicken Bowl']
chipo10
```

```
      order_id  ...  item_price
4            2  ...       16.98
5            3  ...       10.98
13           7  ...       11.25
19          10  ...        8.75
26          13  ...        8.49
...        ...  ...         ...
4590      1825  ...       11.25
4591      1825  ...        8.75
4595      1826  ...        8.75
4599      1827  ...        8.75
4604      1828  ...        8.75

[726 rows x 5 columns]
```

```
chipo_filter = chipo.drop_duplicates(['item_name', 'quantity'])
chipo_filter

chipo_one_prod = chipo_filter[chipo_filter.quantity == 1]
chipo_one_prod

price_per_item = chipo_one_prod[['item_name', 'item_price']]
price_per_item

price_per_item.sort_values(by = "item_price", ascending = False)

chipo.item_name.sort_values()

chipo.sort_values(by='item_name')

chipo.sort_values(by = "item_price", ascending = False).head(1)
```

```
      order_id  quantity  ... choice_description item_price
3598      1443        15  ...                NaN      44.25

[1 rows x 5 columns]
```

```
chipo_salad = chipo[chipo.item_name == "Veggie Salad Bowl"]
chipo_salad
len(chipo_salad)
```

```
18
```

```
chipo_drink_steak_bowl = chipo[(chipo.item_name == "Canned Soda") &
(chipo.quantity > 1)]
chipo_drink_steak_bowl
len(chipo_drink_steak_bowl)
```

```
20
```

```
euro12 = pd.read_csv("Euro2012TEAM.csv")
euro12
```

```
euro12.columns
```

```
euro12[['Team','Goals']]
```

```
discipline = euro12[['Team', 'Yellow Cards', 'Red Cards']]
discipline
```

```
discipline.sort_values(['Red Cards','Yellow Cards'], ascending=True)
```

```
euro12[euro12.Goals > 5]
```

```
euro12[euro12.Goals < 5]
```

```
euro12[euro12.Team.str.startswith('G')]
```

```
      Team  Goals  Shots on target  ...  Subs on Subs off Players Used
5  Germany     10               32  ...       15       15           17
6   Greece      5                8  ...       12       12           20

[2 rows x 35 columns]
```

```
euro12.iloc[:, 0:7]
```

```
euro12.loc[:, 'Team':'Shooting Accuracy']
```

```
euro12.iloc[:,:-3]
```

```
users
```

```
         age gender     occupation zip_code
user_id                                    
1         24      M     technician    85711
2         53      F          other    94043
3         23      M         writer    32067
4         24      M     technician    43537
5         33      F          other    15213
...      ...    ...            ...      ...
939       26      F        student    33319
940       32      M  administrator    02215
941       20      M        student    97229
942       48      F      librarian    78209
943       22      M        student    77841

[943 rows x 4 columns]
```

```
users.groupby('occupation').age.mean()
```

```
def gender_to_numeric(x):
    if x == 'M':
        return 1
    if x == 'F':
        return 0

users['gender_n'] = users['gender'].apply(gender_to_numeric)

users
```

```
         age gender     occupation zip_code  gender_n
user_id
1         24      M     technician    85711         1
2         53      F          other    94043         0
3         23      M         writer    32067         1
4         24      M     technician    43537         1
5         33      F          other    15213         0
...      ...    ...            ...      ...       ...
939       26      F        student    33319         0
940       32      M  administrator    02215         1
941       20      M        student    97229         1
942       48      F      librarian    78209         0
943       22      M        student    77841         1

[943 rows x 5 columns]
```

```
a = users.groupby('occupation').gender_n.sum() /
users.occupation.value_counts() * 100

a

a.sort_values(ascending = False)
```

```
doctor           100.000000
engineer          97.014925
technician        96.296296
retired           92.857143
programmer        90.909091
executive         90.625000
scientist         90.322581
entertainment     88.888889
lawyer            83.333333
salesman          75.000000
educator          72.631579
student           69.387755
other             65.714286
marketing         61.538462
writer            57.777778
none              55.555556
administrator     54.430380
artist            53.571429
```

```
librarian         43.137255
healthcare        31.250000
homemaker         14.285714
dtype: float64
```

```
users.groupby('occupation').age.agg(['min', 'max'])
```

```
users.groupby(['occupation', 'gender']).age.mean()
```

```
gender_ocup = users.groupby(['occupation', 'gender']).agg({'gender':
'count'})
```

```
occup_count = users.groupby(['occupation']).agg('count')
```

```
occup_gender = gender_ocup.div(occup_count, level = "occupation") *
100
```

```
occup_gender.loc[: , 'gender']
```

```
url =
"https://raw.githubusercontent.com/guipsamora/pandas_exercises/master/
04_Apply/US_Crime_Rates/US_Crime_Rates_1960_2014.csv"
crime = pd.read_csv(url)
crime.head()
```

```
   Year  Population    Total  ...  Burglary  Larceny_Theft
Vehicle_Theft
0  1960   179323175  3384200  ...    912100        1855400
328200
1  1961   182992000  3488000  ...    949600        1913000
336000
2  1962   185771000  3752200  ...    994300        2089600
366800
3  1963   188483000  4109500  ...   1086400        2297800
408300
4  1964   191141000  4564600  ...   1213200        2514400
472800

[5 rows x 12 columns]
```

```
crime.info
```

```
<bound method DataFrame.info of     Year  Population     Total  ...
Burglary  Larceny_Theft  Vehicle_Theft
0   1960   179323175   3384200  ...    912100        1855400
328200
1   1961   182992000   3488000  ...    949600        1913000
336000
2   1962   185771000   3752200  ...    994300        2089600
366800
3   1963   188483000   4109500  ...   1086400        2297800
408300
```

```
4   1964   191141000   4564600  ...   1213200         2514400
472800
5   1965   193526000   4739400  ...   1282500         2572600
496900
6   1966   195576000   5223500  ...   1410100         2822000
561200
7   1967   197457000   5903400  ...   1632100         3111600
659800
8   1968   199399000   6720200  ...   1858900         3482700
783600
9   1969   201385000   7410900  ...   1981900         3888600
878500
10  1970   203235298   8098000  ...   2205000         4225800
928400
11  1971   206212000   8588200  ...   2399300         4424200
948200
12  1972   208230000   8248800  ...   2375500         4151200
887200
13  1973   209851000   8718100  ...   2565500         4347900
928800
14  1974   211392000  10253400  ...   3039200         5262500
977100
15  1975   213124000  11292400  ...   3265300         5977700
1009600
16  1976   214659000  11349700  ...   3108700         6270800
966000
17  1977   216332000  10984500  ...   3071500         5905700
977700
18  1978   218059000  11209000  ...   3128300         5991000
1004100
19  1979   220099000  12249500  ...   3327700         6601000
1112800
20  1980   225349264  13408300  ...   3795200         7136900
1131700
21  1981   229146000  13423800  ...   3779700         7194400
1087800
22  1982   231534000  12974400  ...   3447100         7142500
1062400
23  1983   233981000  12108600  ...   3129900         6712800
1007900
24  1984   236158000  11881800  ...   2984400         6591900
1032200
25  1985   238740000  12431400  ...   3073300         6926400
1102900
26  1986   240132887  13211869  ...   3241410         7257153
1224137
27  1987   242282918  13508700  ...   3236184         7499900
1288674
28  1988   245807000  13923100  ...   3218100         7705900
1432900
```

```
29  1989   248239000  14251400  ...   3168200        7872400
1564800
30  1990   248709873  14475600  ...   3073900        7945700
1635900
31  1991   252177000  14872900  ...   3157200        8142200
1661700
32  1992   255082000  14438200  ...   2979900        7915200
1610800
33  1993   257908000  14144800  ...   2834800        7820900
1563100
34  1994   260341000  13989500  ...   2712800        7879800
1539300
35  1995   262755000  13862700  ...   2593800        7997700
1472400
36  1996   265228572  13493863  ...   2506400        7904700
1394200
37  1997   267637000  13194571  ...   2460526        7743760
1354189
38  1998   270296000  12475634  ...   2329950        7373886
1240754
39  1999   272690813  11634378  ...   2100739        6955520
1152075
40  2000   281421906  11608072  ...   2050992        6971590
1160002
41  2001   285317559  11876669  ...   2116531        7092267
1228391
42  2002   287973924  11878954  ...   2151252        7057370
1246646
43  2003   290690788  11826538  ...   2154834        7026802
1261226
44  2004   293656842  11679474  ...   2144446        6937089
1237851
45  2005   296507061  11565499  ...   2155448        6783447
1235859
46  2006   299398484  11401511  ...   2183746        6607013
1192809
47  2007   301621157  11251828  ...   2176140        6568572
1095769
48  2008   304374846  11160543  ...   2228474        6588046
958629
49  2009   307006550  10762956  ...   2203313        6338095
795652
50  2010   309330219  10363873  ...   2168457        6204601
739565
51  2011   311587816  10258774  ...   2185140        6151095
716508
52  2012   313873685  10219059  ...   2109932        6168874
723186
53  2013   316497531   9850445  ...   1931835        6018632
700294
```

```
54  2014   318857056   9475816  ...   1729806        5858496
689527

[55 rows x 12 columns]>
```

```
crime.describe()
```

```
             Year    Population  ...  Larceny_Theft  Vehicle_Theft
count    55.00000  5.500000e+01  ...   5.500000e+01   5.500000e+01
mean   1987.00000  2.461556e+08  ...   5.959947e+06   1.028614e+06
std      16.02082  4.166216e+07  ...   1.846401e+06   3.455693e+05
min    1960.00000  1.793232e+08  ...   1.855400e+06   3.282000e+05
25%    1973.50000  2.106215e+08  ...   4.843350e+06   7.896260e+05
50%    1987.00000  2.422829e+08  ...   6.591900e+06   1.032200e+06
75%    2000.50000  2.833697e+08  ...   7.168450e+06   1.239302e+06
max    2014.00000  3.188571e+08  ...   8.142200e+06   1.661700e+06

[8 rows x 12 columns]
```

```
crime.Year = pd.to_datetime(crime.Year, format='%Y')
crime
```

```
crime = crime.set_index('Year', drop = True)
crime.head()
```

```
            Population    Total  ...  Larceny_Theft  Vehicle_Theft
Year                             ...
1960-01-01   179323175  3384200  ...        1855400         328200
1961-01-01   182992000  3488000  ...        1913000         336000
1962-01-01   185771000  3752200  ...        2089600         366800
1963-01-01   188483000  4109500  ...        2297800         408300
1964-01-01   191141000  4564600  ...        2514400         472800

[5 rows x 11 columns]
```

```
del crime['Total']
crime.head()
```

```
            Population  Violent  ...  Larceny_Theft  Vehicle_Theft
Year                             ...
1960-01-01   179323175   288460  ...        1855400         328200
1961-01-01   182992000   289390  ...        1913000         336000
1962-01-01   185771000   301510  ...        2089600         366800
1963-01-01   188483000   316970  ...        2297800         408300
1964-01-01   191141000   364220  ...        2514400         472800

[5 rows x 10 columns]
```

```
crimes = crime.resample('10AS').sum()
crimes
```

```
crime.idxmax(0)
```

```
Population           2014-01-01
Violent              1992-01-01
Property             1991-01-01
Murder               1991-01-01
Forcible_Rape        1992-01-01
Robbery              1991-01-01
Aggravated_assault   1993-01-01
Burglary             1980-01-01
Larceny_Theft        1991-01-01
Vehicle_Theft        1991-01-01
dtype: datetime64[ns]
```