Module 6: GroupBy

- Creates a category/chunk/segment from a DataFrame
- Split data into unique groups from the variable/column of choice.

In [1]:

```
import pandas as pd
fortune= pd.read_csv("data/fortune1000.csv", index_col="Rank")
fortune.head()
```

Out[1]:

Rank	Company	Sector	Industry	Location	Revenue	Profits	Employees
1	Walmart	Retailing	General Merchandisers	Bentonville, AR	482130	14694	2300000
2	Exxon Mobil	Energy	Petroleum Refining	Irving, TX	246204	16150	75600
3	Apple	Technology	Computers, Office Equipment	Cupertino, CA	233715	53394	110000
4	Berkshire Hathaway	Financials	Insurance: Property and Casualty (Stock)	Omaha, NE	210821	24083	331000
5	McKesson	Health Care	Wholesalers: Health Care	San Francisco, CA	181241	1476	70400

- by using **groupby()** method, Pandas will create a new object called *groupby.DataFrameGroupBy* object.
- **groupby object** only works when we apply a method on it.

In [2]:

```
sectors = fortune.groupby("Sector")
sectors
```

Out[2]:

```
<pandas.core.groupby.DataFrameGroupBy object at 0x000001DD97C9BDD8>
```

In [3]:

```
sectors.groups.keys()
```

Out[3]:

```
dict_keys(['Aerospace & Defense', 'Apparel', 'Business Services', 'Chemical
s', 'Energy', 'Engineering & Construction', 'Financials', 'Food and Drug Sto
res', 'Food, Beverages & Tobacco', 'Health Care', 'Hotels, Resturants & Leis
ure', 'Household Products', 'Industrials', 'Materials', 'Media', 'Motor Vehi
cles & Parts', 'Retailing', 'Technology', 'Telecommunications', 'Transportat
ion', 'Wholesalers'])
```

Operations and Attributes with groupby Object

- **size()** : Number of rows in each group.
- **first()** : Compute first of values within each group.
- **last()** : Compute last of group values.

In [4]:

```
sectors.size()
```

Out[4]:

```
Sector
Aerospace & Defense             20
Apparel                         15
Business Services               51
Chemicals                       30
Energy                         122
Engineering & Construction      26
Financials                     139
Food and Drug Stores            15
Food, Beverages & Tobacco       43
Health Care                     75
Hotels, Resturants & Leisure    25
Household Products              28
Industrials                     46
Materials                       43
Media                           25
Motor Vehicles & Parts          24
Retailing                       80
Technology                     102
Telecommunications              15
Transportation                  36
Wholesalers                     40
dtype: int64
```

The first value in the Aerospace & Defense Sector is the Company, **Boeing**. If we see in the list of Sector column, the first company is **Boeing** as well.

In [5]:

```
sectors.first().head(2)
```

Out[5]:

Sector	Company	Industry	Location	Revenue	Profits	Employees
Aerospace & Defense	Boeing	Aerospace and Defense	Chicago, IL	96114	5176	161400
Apparel	Nike	Apparel	Beaverton, OR	30601	3273	62600

In [6]:

```
fortune[fortune["Sector"] == "Aerospace & Defense"].head(2)
```

Out[6]:

Rank	Company	Sector	Industry	Location	Revenue	Profits	Employees
24	Boeing	Aerospace & Defense	Aerospace and Defense	Chicago, IL	96114	5176	161400
45	United Technologies	Aerospace & Defense	Aerospace and Defense	Farmington, CT	61047	7608	197200

The last Company in Aerospace & Defense Sector is **Delta Tucker Holdings**. If we go through the original DataFrame, we can see that the last company is also **Delta Tucker Holdings**.

In [7]:

```
sectors.last().head(2)
```

Out[7]:

Sector	Company	Industry	Location	Revenue	Profits	Employees
Aerospace & Defense	Delta Tucker Holdings	Aerospace and Defense	McLean, VA	1923	-133	12000
Apparel	Guess	Apparel	Los Angeles, CA	2204	82	13500

In [8]:

```
fortune[fortune["Sector"] == "Aerospace & Defense"].tail(1)
```

Out[8]:

Rank	Company	Sector	Industry	Location	Revenue	Profits	Employees
987	Delta Tucker Holdings	Aerospace & Defense	Aerospace and Defense	McLean, VA	1923	-133	12000

Attributes: groups

Returns a dictionary which contains the index of values belonging to each group.

In [9]:

```
sectors.groups
```

Out[9]:

```
{'Aerospace & Defense': Int64Index([ 24,  45,  60,  88, 118, 120, 209, 24
5, 282, 378, 389, 490, 560,
             605, 785, 788, 836, 903, 958, 987],
            dtype='int64', name='Rank'),
 'Apparel': Int64Index([91, 231, 340, 354, 448, 547, 575, 597, 683, 695, 7
26, 794, 877,
             882, 917],
            dtype='int64', name='Rank'),
 'Business Services': Int64Index([144, 186, 199, 204, 221, 248, 249, 294,
307, 312, 355, 392, 404,
             440, 467, 468, 481, 485, 492, 503, 545, 626, 635, 652, 677, 6
94,
             714, 729, 734, 735, 737, 744, 767, 776, 777, 783, 791, 792, 7
96,
             801, 803, 816, 819, 820, 869, 870, 886, 939, 951, 952, 993],
            dtype='int64', name='Rank'),
 'Chemicals': Int64Index([ 56, 101, 182, 189, 206, 253, 262, 277, 288, 29
6, 316, 538, 549,
             555, 566, 580, 613, 624, 654, 668, 717, 720, 724, 758, 761, 8
29,
             865, 898, 934, 949],
            dtype='int64', name='Rank'),
 'Energy': Int64Index([  2,  14,  30,  32,  42,  65,  90,  95,  98, 104,
             ...
             953, 955, 962, 966, 980, 981, 983, 995, 997, 997],
            dtype='int64', name='Rank', length=122),
 'Engineering & Construction': Int64Index([155, 156, 235, 260, 301, 314, 3
52, 381, 433, 478, 498, 501, 516,
             572, 576, 640, 719, 738, 822, 863, 874, 892, 929, 963, 985, 9
94],
            dtype='int64', name='Rank'),
 'Financials': Int64Index([  4,  16,  23,  26,  27,  29,  35,  40,  43,  4
9,
             ...
             920, 923, 925, 926, 936, 938, 959, 976, 988, 996],
            dtype='int64', name='Rank', length=139),
 'Food and Drug Stores': Int64Index([7, 17, 19, 87, 107, 160, 181, 595, 61
4, 631, 669, 756, 809, 849,
             928],
            dtype='int64', name='Rank'),
 'Food, Beverages & Tobacco': Int64Index([ 41,  44,  62,  66,  84,  94, 10
6, 149, 153, 161, 176, 207, 215,
             242, 266, 304, 336, 337, 362, 397, 413, 429, 452, 456, 460, 5
35,
             542, 567, 574, 587, 608, 615, 636, 664, 686, 702, 770, 787, 7
95,
             804, 889, 902, 954],
            dtype='int64', name='Rank'),
 'Health Care': Int64Index([  5,   6,  12,  21,  22,  33,  39,  46,  52,
55,  63,  72,  79,
              86, 123, 124, 125, 130, 138, 140, 141, 168, 172, 200, 201, 2
02,
             263, 268, 278, 286, 287, 290, 291, 305, 325, 358, 359, 372, 4
20,
             430, 431, 447, 457, 465, 469, 515, 527, 537, 553, 559, 583, 6
```

```
18,
             629, 661, 684, 693, 697, 708, 746, 749, 766, 775, 790, 799, 8
15,
             818, 845, 868, 881, 883, 935, 960, 965, 977, 978],
            dtype='int64', name='Rank'),
 'Hotels, Resturants & Leisure': Int64Index([109, 146, 195, 218, 241, 254,
309, 371, 444, 466, 534, 546, 557,
             562, 585, 731, 763, 764, 779, 797, 895, 915, 918, 941, 999],
            dtype='int64', name='Rank'),
 'Household Products': Int64Index([ 34, 151, 174, 256, 261, 328, 338, 345,
370, 434, 441, 450, 455,
             533, 550, 554, 601, 665, 699, 716, 856, 890, 900, 907, 932, 9
50,
             967, 992],
            dtype='int64', name='Rank'),
 'Industrials': Int64Index([  11,   59,   75,   93,   97,  128,  134,  14
8,  211,  224,  360,
              377,  396,  412,  419,  539,  544,  596,  622,  623,  630,
645,
              655,  658,  666,  674,  690,  757,  782,  789,  806,  813,
823,
              831,  832,  835,  841,  847,  866,  867,  873,  887,  942,
947,
              964, 1000],
            dtype='int64', name='Rank'),
 'Materials': Int64Index([126, 127, 170, 244, 251, 303, 321, 341, 356, 37
3, 375, 383, 417,
             418, 426, 435, 446, 480, 489, 514, 518, 577, 586, 616, 617, 6
20,
             625, 628, 637, 639, 641, 660, 667, 687, 691, 751, 774, 793, 8
57,
             885, 911, 982, 986],
            dtype='int64', name='Rank'),
 'Media': Int64Index([ 53,  96,  99, 203, 213, 255, 327, 366, 406, 414, 52
5, 530, 681,
             705, 707, 722, 741, 752, 759, 762, 821, 913, 968, 969, 989],
            dtype='int64', name='Rank'),
 'Motor Vehicles & Parts': Int64Index([  8,   9,  70, 147, 154, 169, 281,
310, 334, 339, 424, 428, 470,
             588, 591, 604, 647, 670, 718, 739, 811, 957, 961, 979],
            dtype='int64', name='Rank'),
 'Retailing': Int64Index([  1,  15,  28,  38,  47,  71,  89, 103, 111, 13
2, 136, 139, 143,
             145, 177, 180, 191, 196, 197, 228, 234, 237, 238, 240, 258, 2
67,
             269, 280, 293, 297, 299, 302, 330, 342, 346, 361, 365, 374, 3
79,
             380, 393, 415, 427, 439, 495, 500, 502, 511, 512, 517, 523, 5
82,
             592, 602, 607, 610, 621, 643, 644, 648, 657, 675, 689, 721, 7
48,
             771, 773, 807, 812, 824, 825, 827, 840, 876, 894, 899, 922, 9
37,
             940, 973],
            dtype='int64', name='Rank'),
 'Technology': Int64Index([  3,  18,  20,  25,  31,  36,  51,  54,  77, 11
0,
             ...
             931, 943, 944, 946, 956, 970, 971, 975, 984, 990],
            dtype='int64', name='Rank', length=102),
 'Telecommunications': Int64Index([10, 13, 37, 116, 159, 187, 292, 333, 39
```

```
9, 443, 461, 496, 526, 619,
            786],
           dtype='int64', name='Rank'),
 'Transportation': Int64Index([ 48,  58,  67,  68,  80, 129, 142, 208, 23
9, 270, 353, 390, 395,
            405, 407, 416, 432, 459, 521, 528, 570, 578, 642, 672, 709, 7
13,
            740, 800, 814, 858, 884, 930, 933, 945, 972, 974],
           dtype='int64', name='Rank'),
 'Wholesalers': Int64Index([ 57,  64,  92, 102, 108, 119, 122, 167, 183, 1
85, 212, 276, 285,
            315, 317, 320, 323, 335, 351, 357, 369, 391, 423, 463, 474, 4
77,
            484, 564, 581, 599, 627, 653, 685, 727, 747, 780, 808, 837, 8
75,
            991],
           dtype='int64', name='Rank')}
```

get_group() method

Construct DataFrame of a specific group by specifying its name

For example, , we want to have a list of companies in the Retailing Sector.

In [10]:

```
sectors.get_group("Retailing").sort_values(by="Revenue", ascending=False).head(10)
```

Out[10]:

Rank	Company	Employees	Industry	Location	Profits	Revenue
1	Walmart	2300000	General Merchandisers	Bentonville, AR	14694	482130
15	Costco	161000	Specialty Retailers: Other	Issaquah, WA	2377	116199
28	Home Depot	385000	Specialty Retailers: Other	Atlanta, GA	7009	88519
38	Target	341000	General Merchandisers	Minneapolis, MN	3363	73785
47	Lowe’s	225000	Specialty Retailers: Other	Mooresville, NC	2546	59074
71	Best Buy	125000	Specialty Retailers: Other	Richfield, MN	897	39745
89	TJX	216000	Specialty Retailers: Apparel	Framingham, MA	2278	30945
103	Macy’s	157500	General Merchandisers	Cincinnati, OH	1072	27079
111	Sears Holdings	178000	General Merchandisers	Hoffman Estates, IL	-1129	25146
132	Staples	58963	Specialty Retailers: Other	Framingham, MA	379	21059

We can also use **masking technique**. The approach is different but the output is the same.

In [11]:

```
fortune[fortune["Sector"] == "Retailing"]
```

Out[11]:

Rank	Company	Sector	Industry	Location	Revenue	Profits	Employees
1	Walmart	Retailing	General Merchandisers	Bentonville, AR	482130	14694	2300000
15	Costco	Retailing	Specialty Retailers: Other	Issaquah, WA	116199	2377	161000
28	Home Depot	Retailing	Specialty Retailers: Other	Atlanta, GA	88519	7009	385000
38	Target	Retailing	General Merchandisers	Minneapolis, MN	73785	3363	341000
47	Lowe’s	Retailing	Specialty Retailers: Other	Mooresville, NC	59074	2546	225000
71	Best Buy	Retailing	Specialty Retailers: Other	Richfield, MN	39745	897	125000
89	TJX	Retailing	Specialty Retailers: Apparel	Framingham, MA	30945	2278	216000
103	Macy’s	Retailing	General Merchandisers	Cincinnati, OH	27079	1072	157500
111	Sears Holdings	Retailing	General Merchandisers	Hoffman Estates, IL	25146	-1129	178000
132	Staples	Retailing	Specialty Retailers: Other	Framingham, MA	21059	379	58963
136	AutoNation	Retailing	Automotive Retailing, Services	Fort Lauderdale, FL	20862	443	26000
139	Dollar General	Retailing	General Merchandisers	Goodlettsville, TN	20369	1165	113400
143	Penske Automotive Group	Retailing	Automotive Retailing, Services	Bloomfield Hills, MI	19361	326	23000
145	Kohl’s	Retailing	General Merchandisers	Menomonee Falls, WI	19204	673	86000
177	Gap	Retailing	Specialty Retailers: Apparel	San Francisco, CA	15797	920	141000
180	Dollar Tree	Retailing	Specialty Retailers: Other	Chesapeake, VA	15498	282	111550
191	CarMax	Retailing	Automotive Retailing, Services	Richmond, VA	14874	597	22064
196	Office Depot	Retailing	Specialty Retailers: Other	Boca Raton, FL	14485	8	49000
197	Nordstrom	Retailing	General Merchandisers	Seattle, WA	14437	600	72500
228	J.C. Penney	Retailing	General Merchandisers	Plano, TX	12625	-513	105000

Rank	Company	Sector	Industry	Location	Revenue	Profits	Employees
234	L Brands	Retailing	Specialty Retailers: Apparel	Columbus, OH	12154	1253	55400
237	Ross Stores	Retailing	Specialty Retailers: Apparel	Dublin, CA	11940	1021	77800
238	Bed Bath & Beyond	Retailing	Specialty Retailers: Other	Union, NJ	11881	958	60000
240	Toys “R” Us	Retailing	Specialty Retailers: Other	Wayne, NJ	11802	-130	62000
258	Murphy USA	Retailing	Specialty Retailers: Other	El Dorado, AR	10885	176	7100
267	Group 1 Automotive	Retailing	Automotive Retailing, Services	Houston, TX	10633	94	12886
269	Hertz Global Holdings	Retailing	Automotive Retailing, Services	Estero, FL	10535	273	30000
280	AutoZone	Retailing	Specialty Retailers: Other	Memphis, TN	10187	1160	63990
293	Advance Auto Parts	Retailing	Specialty Retailers: Other	Roanoke, VA	9737	473	56500
297	Sonic Automotive	Retailing	Automotive Retailing, Services	Charlotte, NC	9624	86	9800
...	...	...	...	...	...	...	...
523	Ascena Retail Group	Retailing	Specialty Retailers: Apparel	Mahwah, NJ	4803	-237	31000
582	Tiffany	Retailing	Specialty Retailers: Other	New York, NY	4105	464	12200
592	Cabela’s	Retailing	Specialty Retailers: Other	Sidney, NE	3998	189	19700
602	Ulta Salon Cosmetics & Fragrance	Retailing	Specialty Retailers: Other	Bolingbrook, IL	3924	320	18100
607	Fastenal	Retailing	Specialty Retailers: Other	Winona, MN	3869	516	20746
610	Sally Beauty Holdings	Retailing	Specialty Retailers: Other	Denton, TX	3834	235	21033
621	HSN	Retailing	Specialty Retailers: Other	St. Petersburg, FL	3691	169	6600
643	American Eagle Outfitters	Retailing	Specialty Retailers: Apparel	Pittsburgh, PA	3522	218	22150
644	Abercrombie & Fitch	Retailing	Specialty Retailers: Apparel	New Albany, OH	3519	36	28500
648	Tailored Brands	Retailing	Specialty Retailers: Apparel	Houston, TX	3496	-1027	21250

Rank	Company	Sector	Industry	Location	Revenue	Profits	Employees
657	Urban Outfitters	Retailing	Specialty Retailers: Apparel	Philadelphia, PA	3445	225	16680
675	Rent-A-Center	Retailing	Specialty Retailers: Other	Plano, TX	3278	-867	24300
689	Aaron’s	Retailing	Specialty Retailers: Other	Atlanta, GA	3180	136	12700
721	Genesco	Retailing	Specialty Retailers: Apparel	Nashville, TN	3022	95	18363
748	Systemax	Retailing	Specialty Retailers: Other	Port Washington, NY	2908	-100	3300
771	PriceSmart	Retailing	Specialty Retailers: Other	San Diego, CA	2803	89	7592
773	Bon-Ton Stores	Retailing	General Merchandisers	York, PA	2790	-57	24100
807	Chico’s FAS	Retailing	Specialty Retailers: Apparel	Fort Myers, FL	2642	2	14755
812	DSW	Retailing	Specialty Retailers: Apparel	Columbus, OH	2620	136	11900
824	Caleres	Retailing	Specialty Retailers: Apparel	St. Louis, MO	2577	82	11000
825	PC Connection	Retailing	Specialty Retailers: Other	Merrimack, NH	2574	47	2155
827	Mattress Firm Holding	Retailing	Specialty Retailers: Other	Houston, TX	2547	65	7186
840	J.Crew Group	Retailing	Specialty Retailers: Apparel	New York, NY	2506	-1243	10450
876	Express	Retailing	Specialty Retailers: Apparel	Columbus, OH	2350	117	10710
894	Party City Holdco	Retailing	Specialty Retailers: Other	Elmsford, NY	2295	11	12888
899	Sears Hometown & Outlet Stores	Retailing	Specialty Retailers: Other	Hoffman Estates, IL	2288	-27	2918
922	Outerwall	Retailing	Specialty Retailers: Other	Bellevue, WA	2195	44	2670
937	hhgregg	Retailing	Specialty Retailers: Other	Indianapolis, IN	2129	-133	4941
940	Restoration Hardware	Retailing	Specialty Retailers: Other	Corte Madera, CA	2109	91	4200
973	99 Cents Only Stores	Retailing	Specialty Retailers: Other	Commerce, CA	1999	-232	18200

80 rows × 7 columns

sum() method

returns a summation on each category for each other Numeric Columns.

In [12]:

```
sectors.sum()
```

Out[12]:

Sector	Revenue	Profits	Employees
Aerospace & Defense	357940	28742	968057
Apparel	95968	8236	346397
Business Services	272195	28227	1361050
Chemicals	243897	22628	463651
Energy	1517809	-73447	1188927
Engineering & Construction	153983	5304	406708
Financials	2217159	260209	3359948
Food and Drug Stores	483769	16759	1395398
Food, Beverages & Tobacco	555967	51417	1211632
Health Care	1614707	106114	2678289
Hotels, Resturants & Leisure	169546	20697	2484245
Household Products	234737	14428	646038
Industrials	497581	20764	1545229
Materials	259145	4428	638123
Media	220764	24347	550314
Motor Vehicles & Parts	482540	25898	1082560
Retailing	1465076	47830	6227629
Technology	1377600	180473	3578949
Telecommunications	461834	48637	832468
Transportation	408508	44169	1536793
Wholesalers	444800	8233	525597

count() method

Count data for every unique group. For example, according to the DataFrame below, we can know that there are 20 Employees who are working in Aerospace & Defense sector.

In [13]:

```
sectors.count().head(10)
```

Out[13]:

Sector	Company	Industry	Location	Revenue	Profits	Employees
Aerospace & Defense	20	20	20	20	20	20
Apparel	15	15	15	15	15	15
Business Services	51	51	51	51	51	51
Chemicals	30	30	30	30	30	30
Energy	122	122	122	122	122	122
Engineering & Construction	26	26	26	26	26	26
Financials	139	139	139	139	139	139
Food and Drug Stores	15	15	15	15	15	15
Food, Beverages & Tobacco	43	43	43	43	43	43
Health Care	75	75	75	75	75	75

mean() method

returns average value for each Numeric columns.

In [14]:

```
sectors.mean()
```

Out[14]:

Sector	Revenue	Profits	Employees
Aerospace & Defense	17897.000000	1437.100000	48402.850000
Apparel	6397.866667	549.066667	23093.133333
Business Services	5337.156863	553.470588	26687.254902
Chemicals	8129.900000	754.266667	15455.033333
Energy	12441.057377	-602.024590	9745.303279
Engineering & Construction	5922.423077	204.000000	15642.615385
Financials	15950.784173	1872.007194	24172.287770
Food and Drug Stores	32251.266667	1117.266667	93026.533333
Food, Beverages & Tobacco	12929.465116	1195.744186	28177.488372
Health Care	21529.426667	1414.853333	35710.520000
Hotels, Resturants & Leisure	6781.840000	827.880000	99369.800000
Household Products	8383.464286	515.285714	23072.785714
Industrials	10816.978261	451.391304	33591.934783
Materials	6026.627907	102.976744	14840.069767
Media	8830.560000	973.880000	22012.560000
Motor Vehicles & Parts	20105.833333	1079.083333	45106.666667
Retailing	18313.450000	597.875000	77845.362500
Technology	13505.882353	1769.343137	35087.735294
Telecommunications	30788.933333	3242.466667	55497.866667
Transportation	11347.444444	1226.916667	42688.694444
Wholesalers	11120.000000	205.825000	13139.925000

Extract the average of specific columns by specifying its name inside square brackets at the end.

In [15]:

```
sectors.mean()["Revenue"]
```

Out[15]:

```
Sector
Aerospace & Defense             17897.000000
Apparel                          6397.866667
Business Services                5337.156863
Chemicals                        8129.900000
Energy                          12441.057377
Engineering & Construction       5922.423077
Financials                      15950.784173
Food and Drug Stores            32251.266667
Food, Beverages & Tobacco       12929.465116
Health Care                     21529.426667
Hotels, Resturants & Leisure     6781.840000
Household Products               8383.464286
Industrials                     10816.978261
Materials                        6026.627907
Media                            8830.560000
Motor Vehicles & Parts          20105.833333
Retailing                       18313.450000
Technology                      13505.882353
Telecommunications              30788.933333
Transportation                  11347.444444
Wholesalers                     11120.000000
Name: Revenue, dtype: float64
```

max() method

This method will return the highest values in the specified column for each group.

For example, we can use it when we want to identify which Sectors give the most profit.

In [16]:

```
sectors["Profits"].max()
```

Out[16]:

```
Sector
Aerospace & Defense              7608
Apparel                          3273
Business Services                6328
Chemicals                        7685
Energy                          16150
Engineering & Construction        803
Financials                      24442
Food and Drug Stores             5237
Food, Beverages & Tobacco        7351
Health Care                     18108
Hotels, Resturants & Leisure     5920
Household Products               7036
Industrials                      4833
Materials                         991
Media                            8382
Motor Vehicles & Parts           9687
Retailing                       14694
Technology                      53394
Telecommunications              17879
Transportation                   7610
Wholesalers                      1472
Name: Profits, dtype: int64
```

min() method

This method is the opposite of max() method. It will return the lowest values in the specified column for each group.

Here is an example where we use the min() method to get the least amount of revenue a company received for each sector.

In [17]:

```
sectors["Revenue"].min()
```

Out[17]:

```
Sector
Aerospace & Defense             1923
Apparel                         2204
Business Services               1910
Chemicals                       2084
Energy                          1898
Engineering & Construction      1909
Financials                      1902
Food and Drug Stores            2151
Food, Beverages & Tobacco       2066
Health Care                     1987
Hotels, Resturants & Leisure    1896
Household Products              1914
Industrials                     1895
Materials                       1924
Media                           1921
Motor Vehicles & Parts          1986
Retailing                       1999
Technology                      1920
Telecommunications              2726
Transportation                  1995
Wholesalers                     1917
Name: Revenue, dtype: int64
```

Grouping by multiple groups

In the previous example, we only group by one column/category. Now, let's do it with multiple columns.

In [18]:

```
multiple = fortune.groupby(["Sector", "Industry"])
multiple
```

Out[18]:

```
<pandas.core.groupby.DataFrameGroupBy object at 0x000001DD992666A0>
```

Multiple groups give more detailed information. The return value is more spread out.

In the example, we can see that there are 7 Industries in the Business Services sector. Meanwhile, there is only one industry in sector Aerospace & Defense.

In [19]:

```
multiple.sum()
```

Out[19]:

Sector	Industry	Revenue	Profits	Employees
Aerospace & Defense	**Aerospace and Defense**	357940	28742	968057
Apparel	**Apparel**	95968	8236	346397
Business Services	**Advertising, marketing**	22748	1549	124100
	Diversified Outsourcing Services	64829	4305	708330
	Education	7485	69	46755
	Financial Data Services	100778	17456	264926
	Miscellaneous	11185	2130	37720
	Temporary Help	34716	1000	60020
	Waste Management	30454	1718	119199
Chemicals	**Chemicals**	243897	22628	463651
Energy	**Energy**	67749	-13038	70072
	Mining, Crude-Oil Production	176435	-124555	149110
	Miscellaneous	3159	476	6700
	Oil and Gas Equipment, Services	82827	-4047	225795
	Petroleum Refining	705472	31888	244880
	Pipelines	138756	5814	72141
	Utilities: Gas and Electric	343411	30015	420229
Engineering & Construction	**Engineering, Construction**	99774	1765	368104
	Homebuilders	54209	3539	38604
Financials	**Commercial Banks**	623669	129760	1539024
	Diversified Financials	327075	28548	329114
	Insurance: Life, Health (Mutual)	181336	5601	58801
	Insurance: Life, Health (stock)	245551	19731	202233
	Insurance: Property and Casualty (Mutual)	143682	8031	133148
	Insurance: Property and Casualty (Stock)	548027	44803	794123
	Real estate	62748	9480	160579
	Securities	85071	14255	142926
Food and Drug Stores	**Food and Drug Stores**	483769	16759	1395398
Food, Beverages & Tobacco	**Beverages**	79396	11698	199107
	Food Consumer Products	238954	19263	665439
...	**...**	...	...	...
Materials	**Metals**	99925	-2241	193559
	Miscellaneous	6070	7	52

Sector	Industry	Revenue	Profits	Employees
	Packaging, Containers	105965	4445	333446
Media	Entertainment	186276	25686	383564
	Publishing, Printing	34488	-1339	166750
Motor Vehicles & Parts	Motor Vehicles and Parts	482540	25898	1082560
Retailing	Automotive Retailing, Services	113823	2550	174974
	General Merchandisers	684320	20137	3408100
	Specialty Retailers: Apparel	122975	4583	770543
	Specialty Retailers: Other	543958	20560	1874012
Technology	Computer Peripherals	48950	3975	175259
	Computer Software	179112	26457	388796
	Computers, Office Equipment	353781	58433	468085
	Information Technology Services	166067	16402	985639
	Internet Services and Retailing	259136	22232	430855
	Network and Other Communications Equipment	125697	18907	299310
	Scientific,Photographic and Control Equipment	65832	9346	231501
	Semiconductors and Other Electronic Components	179025	24721	599504
Telecommunications	Telecommunications	461834	48637	832468
Transportation	Airlines	158948	23800	391601
	Mail, Package, and Freight Delivery	105816	5894	664275
	Miscellaneous	2148	227	4200
	Railroads	48554	9005	121844
	Transportation Equipment	23855	2438	59726
	Transportation and Logistics	33237	895	124691
	Trucking, Truck Leasing	35950	1910	170456
Wholesalers	Miscellaneous	8982	17	9200
	Wholesalers: Diversified	176138	5193	233831
	Wholesalers: Electronics and Office Equipment	147906	1857	166661
	Wholesalers: Food and Grocery	111774	1166	115905

79 rows × 3 columns

agg() method

Aggregates using one or more operations over the specified axis.

- **dictionary** : it will do the specified operations on the columns.
- **list** : it will do all operations in the list.
- **combine both list and dictionary** : enable us to do specific operations on specific columns.

In [20]:

```
sectors.agg({
    "Revenue" : "sum",
    "Profits" : "sum",
    "Employees" : "mean"
})
```

Out[20]:

Sector	Revenue	Profits	Employees
Aerospace & Defense	357940	28742	48402.850000
Apparel	95968	8236	23093.133333
Business Services	272195	28227	26687.254902
Chemicals	243897	22628	15455.033333
Energy	1517809	-73447	9745.303279
Engineering & Construction	153983	5304	15642.615385
Financials	2217159	260209	24172.287770
Food and Drug Stores	483769	16759	93026.533333
Food, Beverages & Tobacco	555967	51417	28177.488372
Health Care	1614707	106114	35710.520000
Hotels, Resturants & Leisure	169546	20697	99369.800000
Household Products	234737	14428	23072.785714
Industrials	497581	20764	33591.934783
Materials	259145	4428	14840.069767
Media	220764	24347	22012.560000
Motor Vehicles & Parts	482540	25898	45106.666667
Retailing	1465076	47830	77845.362500
Technology	1377600	180473	35087.735294
Telecommunications	461834	48637	55497.866667
Transportation	408508	44169	42688.694444
Wholesalers	444800	8233	13139.925000

In [21]:

```
sectors.agg(["sum", "mean","size" ])
```

Out[21]:

	Revenue			Profits			Employees	
	sum	mean	size	sum	mean	size	sum	mean
Sector								
Aerospace & Defense	357940	17897.000000	20	28742	1437.100000	20	968057	48402.85
Apparel	95968	6397.866667	15	8236	549.066667	15	346397	23093.13
Business Services	272195	5337.156863	51	28227	553.470588	51	1361050	26687.25
Chemicals	243897	8129.900000	30	22628	754.266667	30	463651	15455.03
Energy	1517809	12441.057377	122	-73447	-602.024590	122	1188927	9745.30
Engineering & Construction	153983	5922.423077	26	5304	204.000000	26	406708	15642.61
Financials	2217159	15950.784173	139	260209	1872.007194	139	3359948	24172.28
Food and Drug Stores	483769	32251.266667	15	16759	1117.266667	15	1395398	93026.53
Food, Beverages & Tobacco	555967	12929.465116	43	51417	1195.744186	43	1211632	28177.48
Health Care	1614707	21529.426667	75	106114	1414.853333	75	2678289	35710.52
Hotels, Resturants & Leisure	169546	6781.840000	25	20697	827.880000	25	2484245	99369.80
Household Products	234737	8383.464286	28	14428	515.285714	28	646038	23072.78
Industrials	497581	10816.978261	46	20764	451.391304	46	1545229	33591.93
Materials	259145	6026.627907	43	4428	102.976744	43	638123	14840.06
Media	220764	8830.560000	25	24347	973.880000	25	550314	22012.56
Motor Vehicles & Parts	482540	20105.833333	24	25898	1079.083333	24	1082560	45106.66
Retailing	1465076	18313.450000	80	47830	597.875000	80	6227629	77845.36
Technology	1377600	13505.882353	102	180473	1769.343137	102	3578949	35087.73
Telecommunications	461834	30788.933333	15	48637	3242.466667	15	832468	55497.86
Transportation	408508	11347.444444	36	44169	1226.916667	36	1536793	42688.69
Wholesalers	444800	11120.000000	40	8233	205.825000	40	525597	13139.92

In [22]:

```
sectors.agg({
    "Revenue" : ["sum","size"],
    "Profits" : "sum",
    "Employees" : "mean"
})
```

Out[22]:

	Revenue		Profits	Employees
	sum	size	sum	mean
Sector				
Aerospace & Defense	357940	20	28742	48402.850000
Apparel	95968	15	8236	23093.133333
Business Services	272195	51	28227	26687.254902
Chemicals	243897	30	22628	15455.033333
Energy	1517809	122	-73447	9745.303279
Engineering & Construction	153983	26	5304	15642.615385
Financials	2217159	139	260209	24172.287770
Food and Drug Stores	483769	15	16759	93026.533333
Food, Beverages & Tobacco	555967	43	51417	28177.488372
Health Care	1614707	75	106114	35710.520000
Hotels, Resturants & Leisure	169546	25	20697	99369.800000
Household Products	234737	28	14428	23072.785714
Industrials	497581	46	20764	33591.934783
Materials	259145	43	4428	14840.069767
Media	220764	25	24347	22012.560000
Motor Vehicles & Parts	482540	24	25898	45106.666667
Retailing	1465076	80	47830	77845.362500
Technology	1377600	102	180473	35087.735294
Telecommunications	461834	15	48637	55497.866667
Transportation	408508	36	44169	42688.694444
Wholesalers	444800	40	8233	13139.925000

Iterating through Groups

Create an empty DataFrame with columns based on the original dataset.

In [23]:

```
df = pd.DataFrame(columns=fortune.columns)
df
```

Out[23]:

Company	Sector	Industry	Location	Revenue	Profits	Employees

Using for loop, we can get all the information related to the highest revenue. (Its location, which company and industry it belongs to).

In [24]:

```
for sector, data  in sectors:
    highest = data.nlargest(1, "Revenue")
    df = df.append(highest)
df.head(10)
```

Out[24]:

	Company	Sector	Industry	Location	Revenue	Profits	Employees
24	Boeing	Aerospace & Defense	Aerospace and Defense	Chicago, IL	96114	5176	161400
91	Nike	Apparel	Apparel	Beaverton, OR	30601	3273	62600
144	ManpowerGroup	Business Services	Temporary Help	Milwaukee, WI	19330	419	27000
56	Dow Chemical	Chemicals	Chemicals	Midland, MI	48778	7685	49495
2	Exxon Mobil	Energy	Petroleum Refining	Irving, TX	246204	16150	75600
155	Fluor	Engineering & Construction	Engineering, Construction	Irving, TX	18114	413	38758
4	Berkshire Hathaway	Financials	Insurance: Property and Casualty (Stock)	Omaha, NE	210821	24083	331000
7	CVS Health	Food and Drug Stores	Food and Drug Stores	Woonsocket, RI	153290	5237	199000
41	Archer Daniels Midland	Food, Beverages & Tobacco	Food Production	Chicago, IL	67702	1849	32300
5	McKesson	Health Care	Wholesalers: Health Care	San Francisco, CA	181241	1476	70400

In [25]:

```
fortune.sort_values("Revenue", ascending=False).head(10)
```

Out[25]:

Rank	Company	Sector	Industry	Location	Revenue	Profits	Empl
1	Walmart	Retailing	General Merchandisers	Bentonville, AR	482130	14694	23
2	Exxon Mobil	Energy	Petroleum Refining	Irving, TX	246204	16150	
3	Apple	Technology	Computers, Office Equipment	Cupertino, CA	233715	53394	1
4	Berkshire Hathaway	Financials	Insurance: Property and Casualty (Stock)	Omaha, NE	210821	24083	3
5	McKesson	Health Care	Wholesalers: Health Care	San Francisco, CA	181241	1476	
6	UnitedHealth Group	Health Care	Health Care: Insurance and Managed Care	Minnetonka, MN	157107	5813	2
7	CVS Health	Food and Drug Stores	Food and Drug Stores	Woonsocket, RI	153290	5237	1
8	General Motors	Motor Vehicles & Parts	Motor Vehicles and Parts	Detroit, MI	152356	9687	2
9	Ford Motor	Motor Vehicles & Parts	Motor Vehicles and Parts	Dearborn, MI	149558	7373	1
10	AT&T	Telecommunications	Telecommunications	Dallas, TX	146801	13345	2