```
import pandas as pd
import numpy as np


# La groupby si utilizza quando abbiamo a che fare con una variabile
categorica e una continua.
# Ad esempio, se abbiamo una variabile categorica A A B A C B A, e una
continua 3,21,10,4,65,33,26, possiamo raggruppare
# in base ad un canone scelto per la variabile continua. Ad esempio,
se volessimo  conoscere la somma della variabile continua
# rispetto alla variabile categorica, otterremo:  variabile
categorica: A B C    variabile continua: Boh sti cavoli
# tanto il concetto è chiaro


mpg = pd.read_csv("mpg.csv")
mpg
```

```
      mpg  cylinders  displacement horsepower  weight  acceleration  \
0    18.0          8         307.0        130    3504          12.0
1    15.0          8         350.0        165    3693          11.5
2    18.0          8         318.0        150    3436          11.0
3    16.0          8         304.0        150    3433          12.0
4    17.0          8         302.0        140    3449          10.5
..    ...        ...           ...        ...     ...           ...
393  27.0          4         140.0         86    2790          15.6
394  44.0          4          97.0         52    2130          24.6
395  32.0          4         135.0         84    2295          11.6
396  28.0          4         120.0         79    2625          18.6
397  31.0          4         119.0         82    2720          19.4

     model_year  origin                       name
0            70       1  chevrolet chevelle malibu
1            70       1          buick skylark 320
2            70       1         plymouth satellite
3            70       1              amc rebel sst
4            70       1                ford torino
..          ...     ...                        ...
393          82       1            ford mustang gl
394          82       2                  vw pickup
395          82       1              dodge rampage
396          82       1                ford ranger
397          82       1                 chevy s-10

[398 rows x 9 columns]
```

```
# Prendiamo model_year come variabile categorica.
```

```
# Vediamo quindi le occorrenze di model_year

mpg["model_year"].value_counts()
```

```
model_year
73    40
78    36
76    34
82    31
75    30
70    29
79    29
80    29
81    29
71    28
72    28
77    28
74    27
Name: count, dtype: int64
```

```
# Quindi, nel 73 abbiamo avuto 40 auto vendute, nel 78 ne abbiamo
vendute 36, etc


# Da quanto si vede, gli anni non sono in fila. Poichè questa è una
variabile categorica, l'ordinamento
# non è importante. Comunque, dato che stiamo parlando di anni,
conviene ordinarli dal più vecchio al più recente


mpg["model_year"].value_counts().sort_index()
```

```
model_year
70    29
71    28
72    28
73    40
74    27
75    30
76    34
77    28
78    36
79    29
80    29
81    29
```

```
82    31
Name: count, dtype: int64
```

```
# Vediamo quindi ora la funzione groupby. Essa mostrerà come le
variabili numeriche vengono raggruppate
# in base al valore di una variabile categorica.
```

```
mpg_group = mpg.groupby("model_year")
mpg_group
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at
0x000001F491CECAD0>
```

```
mpg_group.sum()
```

```
                mpg  cylinders  displacement  \
model_year
70            513.0        196        8161.0
71            595.0        156        5873.0
72            524.0        163        6114.5
73            684.0        255       10275.0
74            613.0        142        4637.0
75            608.0        168        6166.0
76            733.5        192        6725.0
77            654.5        153        5359.0
78            866.2        193        6401.0
79            727.7        169        5994.0
80            977.2        120        3359.0
81            879.7        134        3924.0
82            983.0        130        3995.0

                                                horsepower  weight
\
model_year

70          13016515015014019822021522519017016015022595595...   97811

71          889095?10010510088100165175153150180170175110 7...   83872

72          95805490861651751501531502081551601909715013 01...   90656

73          17515014513715019815015815021522517510510010 08...  136761

74          95?10010067806575100110105140150150140150836 77...   77704

75          95105727217014515014811010511095110110129758 31...   95304
```

```
76          86819279831401501201521001058190526070531007 81...  104677
77          68805896701451101451301101051009818017019014 97...   83926
78          48665270601101401391059585881009010585110120 14...  103025
79          11585889011013012913813515514212515071658080 77...   88605
80          76607065908890907890759275651056548486767 67?67...   70663
81          84849211084586460676562686365657 4?757510074807...   73165
82          88888885849092?74686863708875706767671108592 11...   76060
```

```
            acceleration  origin  \
model_year
70                 375.5      38
71                 424.0      40
72                 423.5      43
73                 572.5      55
74                 437.5      45
75                 481.5      44
76                 542.0      50
77                 432.2      44
78                 569.0      58
79                 458.6      37
80                 491.1      64
81                 472.9      57
82                 515.8      51
```

```
                                                        name
model_year
70          chevrolet chevelle malibubuick skylark 320plym...
71          datsun pl510chevrolet vega 2300toyota coronafo...
72          toyota corona hardtopdodge colt hardtopvolkswa...
73          buick century 350amc matadorchevrolet malibufo...
74          plymouth dusterford maverickamc hornetchevrole...
75          plymouth valiant customchevrolet novamercury m...
76          fiat 131opel 1900capri iidodge coltrenault 12t...
77          honda accord cvccbuick opel isuzu deluxerenaul...
78          volkswagen rabbit custom dieselford fiestamazd...
79          pontiac lemans v6mercury zephyr 6ford fairmont...
80          vw rabbittoyota corolla tercelchevrolet chevet...
81          plymouth reliantbuick skylarkdodge aries wagon...
82          chevrolet cavalierchevrolet cavalier wagonchev...
```

```
# E' uscito fuori uno schifo. Questo perché abbiamo effettuato la
somma rispetto alla variabile categorica
# model_year sia delle variabili numeriche sia delle variabili
categorica. Quello che ci conviene fare
# e' prendere in considerazione solo le variabili numeriche. Come si
fa? Per prima cosa vediamo qual è il tipo
# delle variabili

mpg.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object
 4   weight        398 non-null    int64
 5   acceleration  398 non-null    float64
 6   model_year    398 non-null    int64
 7   origin        398 non-null    int64
 8   name          398 non-null    object
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

```
# A quanto pare non c'è bisogno di cambiare il tipo di alcune
variabili. Dobbiamo quindi prendere in
# considerazione le sole variabili numeriche. Per farlo:


mpg_group = mpg.groupby('model_year')
[mpg.select_dtypes(include='number').columns].sum()
mpg_group
```

```
              mpg  cylinders  displacement  weight  acceleration
model_year  \
model_year

70          513.0        196        8161.0   97811         375.5
2030
71          595.0        156        5873.0   83872         424.0
1988
72          524.0        163        6114.5   90656         423.5
2016
73          684.0        255       10275.0  136761         572.5
2920
```

```
74           613.0          142          4637.0    77704           437.5
1998
75           608.0          168          6166.0    95304           481.5
2250
76           733.5          192          6725.0   104677           542.0
2584
77           654.5          153          5359.0    83926           432.2
2156
78           866.2          193          6401.0   103025           569.0
2808
79           727.7          169          5994.0    88605           458.6
2291
80           977.2          120          3359.0    70663           491.1
2320
81           879.7          134          3924.0    73165           472.9
2349
82           983.0          130          3995.0    76060           515.8
2542
```

```
             origin
model_year
70               38
71               40
72               43
73               55
74               45
75               44
76               50
77               44
78               58
79               37
80               64
81               57
82               51
```

```
# Se fossimo interessati ad una sola variabile, ad esempio
acceleration:
```

```
mpg.groupby('model_year').sum(["acceleration"])
```

```
              mpg  cylinders  displacement  weight  acceleration
origin
model_year

70          513.0        196        8161.0   97811         375.5
38
71          595.0        156        5873.0   83872         424.0
```

```
40
72          524.0         163         6114.5   90656          423.5
43
73          684.0         255        10275.0  136761          572.5
55
74          613.0         142         4637.0   77704          437.5
45
75          608.0         168         6166.0   95304          481.5
44
76          733.5         192         6725.0  104677          542.0
50
77          654.5         153         5359.0   83926          432.2
44
78          866.2         193         6401.0  103025          569.0
58
79          727.7         169         5994.0   88605          458.6
37
80          977.2         120         3359.0   70663          491.1
64
81          879.7         134         3924.0   73165          472.9
57
82          983.0         130         3995.0   76060          515.8
51

# Se volessimo raggruppare il dataframe rispetto a acceleration e
cylinders.

mpg.groupby('model_year').sum(["acceleration","cylinders"])

                mpg  cylinders  displacement  weight  acceleration
origin
model_year

70          513.0         196         8161.0   97811          375.5
38
71          595.0         156         5873.0   83872          424.0
40
72          524.0         163         6114.5   90656          423.5
43
73          684.0         255        10275.0  136761          572.5
55
74          613.0         142         4637.0   77704          437.5
45
75          608.0         168         6166.0   95304          481.5
44
76          733.5         192         6725.0  104677          542.0
50
77          654.5         153         5359.0   83926          432.2
44
78          866.2         193         6401.0  103025          569.0
```

```
58
79          727.7         169         5994.0   88605          458.6
37
80          977.2         120         3359.0   70663          491.1
64
81          879.7         134         3924.0   73165          472.9
57
82          983.0         130         3995.0   76060          515.8
51

# Per valori uguali di acceleration, avremo valori di cylinders.

# Possiamo anche raggruppare le singole variabili. Se, ad esempio,
volessimo calcolare la media
# delle cilindrate per ogni anno:



cylinders_per_year = mpg.groupby('model_year')['cylinders'].mean()
cylinders_per_year

model_year
70    6.758621
71    5.571429
72    5.821429
73    6.375000
74    5.259259
75    5.600000
76    5.647059
77    5.464286
78    5.361111
79    5.827586
80    4.137931
81    4.620690
82    4.193548
Name: cylinders, dtype: float64



# Possiamo considerare anche più di una variabile. Se, ad esempio,
volessimo la media di cylinders
# e weight per anno:

cylinders_weight_per_year = mpg.groupby('model_year')
[['cylinders','weight','acceleration']].mean()
cylinders_weight_per_year

            cylinders       weight  acceleration
model_year
70           6.758621  3372.793103     12.948276
71           5.571429  2995.428571     15.142857
```

```
72           5.821429  3237.714286     15.125000
73           6.375000  3419.025000     14.312500
74           5.259259  2877.925926     16.203704
75           5.600000  3176.800000     16.050000
76           5.647059  3078.735294     15.941176
77           5.464286  2997.357143     15.435714
78           5.361111  2861.805556     15.805556
79           5.827586  3055.344828     15.813793
80           4.137931  2436.655172     16.934483
81           4.620690  2522.931034     16.306897
82           4.193548  2453.548387     16.638710
```

```
# Possiamo anche raggruppare rispetto a più di una colonna.
# Supponiamo di voler calcolare la somma di cylinders e weight
rispetto a
# model_year e origin. In questo caso:
```

```
a = mpg.groupby(['model_year','origin'])
[['cylinders','weight','acceleration']].mean()
a
```

```
                    cylinders       weight  acceleration
model_year origin
70         1         7.636364  3716.500000     11.977273
           2         4.000000  2309.200000     16.500000
           3         4.000000  2251.000000     14.750000
71         1         6.200000  3401.600000     14.575000
           2         4.000000  2024.000000     16.750000
           3         4.000000  1936.000000     16.375000
72         1         6.888889  3682.666667     14.055556
           2         4.000000  2573.200000     18.700000
           3         3.800000  2300.400000     15.400000
73         1         7.241379  3821.448276     13.620690
           2         4.000000  2335.714286     16.428571
           3         4.250000  2397.250000     15.625000
74         1         6.266667  3503.333333     15.966667
           2         4.000000  2139.333333     15.333333
           3         4.000000  2053.000000     17.666667
75         1         6.400000  3533.200000     16.350000
           2         4.000000  2571.166667     15.083333
           3         4.000000  2303.250000     16.000000
76         1         6.363636  3405.409091     15.786364
           2         4.250000  2611.000000     16.050000
           3         4.500000  2217.500000     16.575000
77         1         6.222222  3422.000000     15.238889
           2         4.000000  2138.750000     15.000000
           3         4.166667  2295.833333     16.316667
```

```
78          1         6.000000  3141.136364     15.545455
            2         4.833333  2691.666667     16.233333
            3         4.000000  2221.250000     16.200000
79          1         6.260870  3210.217391     15.243478
            2         4.250000  2693.750000     18.400000
            3         4.000000  1997.500000     17.200000
80          1         4.285714  2822.428571     16.800000
            2         4.111111  2348.000000     18.366667
            3         4.076923  2290.307692     16.015385
81          1         4.923077  2695.000000     16.053846
            2         4.500000  2725.000000     17.500000
            3         4.333333  2269.166667     16.183333
82          1         4.300000  2637.750000     16.670000
            2         4.000000  2055.000000     19.950000
            3         4.000000  2132.777778     15.833333
```

```
# Consideriamo una delle colonne numeriche: Ad esempio, consideriamo la colonna weight,
# Per valori uguali di model_year, avremo valori diversi di origin, e per diversi valori di origin avremo
# valori diversi di weight. Questo sta a significare che tra il primo livello di aggregazione (model_year)
# ed il secondo livello (origin) c'è un AND nascosto.
# Consideriamo quindi un valore di weight, ad esempio 2309.200000. Questo ci dice che weight è pari
# a 2309.200000 quando (where!!!) model_year = 70 AND origin = 2!!.


# Se abbiamo più di un indice, avremo a che fare con un multiindex.

mpg.groupby(['model_year','origin'])
[['cylinders','weight','acceleration']].mean().index
```

```
MultiIndex([(70, 1),
            (70, 2),
            (70, 3),
            (71, 1),
            (71, 2),
            (71, 3),
            (72, 1),
            (72, 2),
            (72, 3),
            (73, 1),
            (73, 2),
            (73, 3),
            (74, 1),
            (74, 2),
            (74, 3),
```

```
            (75, 1),
            (75, 2),
            (75, 3),
            (76, 1),
            (76, 2),
            (76, 3),
            (77, 1),
            (77, 2),
            (77, 3),
            (78, 1),
            (78, 2),
            (78, 3),
            (79, 1),
            (79, 2),
            (79, 3),
            (80, 1),
            (80, 2),
            (80, 3),
            (81, 1),
            (81, 2),
            (81, 3),
            (82, 1),
            (82, 2),
            (82, 3)],
           names=['model_year', 'origin'])


# CVD


# POSSIAMO ANCHE MOSTRARE UNA SOLA COLONNA CORRISPONDENTE AI NOSTRI
INDICI. AD ESEMPIO, CONSIDERIAMO LA
# SOLITA COLONNA WEIGHT.


 mpg.groupby(['model_year','origin'])['weight'].mean()
```

```
model_year  origin
70          1         3716.500000
            2         2309.200000
            3         2251.000000
71          1         3401.600000
            2         2024.000000
            3         1936.000000
72          1         3682.666667
            2         2573.200000
            3         2300.400000
73          1         3821.448276
```

```
             2          2335.714286
             3          2397.250000
74           1          3503.333333
             2          2139.333333
             3          2053.000000
75           1          3533.200000
             2          2571.166667
             3          2303.250000
76           1          3405.409091
             2          2611.000000
             3          2217.500000
77           1          3422.000000
             2          2138.750000
             3          2295.833333
78           1          3141.136364
             2          2691.666667
             3          2221.250000
79           1          3210.217391
             2          2693.750000
             3          1997.500000
80           1          2822.428571
             2          2348.000000
             3          2290.307692
81           1          2695.000000
             2          2725.000000
             3          2269.166667
82           1          2637.750000
             2          2055.000000
             3          2132.777778
Name: weight, dtype: float64
```

```
# CALCOLO STATISTICHE DI BASE IN SEGUITO A RAGGRUPPAMENTO
```

```
mpg.groupby('model_year').describe().transpose()
```

```
model_year                    70           71           72           73
\
mpg          count    29.000000    28.000000    28.000000    40.000000

             mean     17.689655    21.250000    18.714286    17.100000

             std       5.339231     6.591942     5.435529     4.700245

             min       9.000000    12.000000    11.000000    11.000000
```

	25%	14.000000	15.500000	13.750000	13.000000
	50%	16.000000	19.000000	18.500000	16.000000
	75%	22.000000	27.000000	23.000000	20.000000
	max	27.000000	35.000000	28.000000	29.000000
cylinders	count	29.000000	28.000000	28.000000	40.000000
	mean	6.758621	5.571429	5.821429	6.375000
	std	1.724926	1.665079	2.073708	1.807215
	min	4.000000	4.000000	3.000000	3.000000
	25%	6.000000	4.000000	4.000000	4.000000
	50%	8.000000	6.000000	4.000000	7.000000
	75%	8.000000	6.500000	8.000000	8.000000
	max	8.000000	8.000000	8.000000	8.000000
displacement	count	29.000000	28.000000	28.000000	40.000000
	mean	281.413793	209.750000	218.375000	256.875000
	std	124.421380	115.102410	123.781964	121.722085
	min	97.000000	71.000000	70.000000	68.000000
	25%	198.000000	97.750000	109.250000	121.750000
	50%	307.000000	228.500000	131.000000	276.000000
	75%	383.000000	273.000000	326.000000	350.250000
	max	455.000000	400.000000	429.000000	455.000000
weight	count	29.000000	28.000000	28.000000	40.000000
	mean	3372.793103	2995.428571	3237.714286	3419.025000
	std	852.868663	1061.830859	974.520960	974.809133
	min	1835.000000	1613.000000	2100.000000	1867.000000
	25%	2648.000000	2110.750000	2285.500000	2554.500000
	50%	3449.000000	2798.000000	2956.000000	3338.500000

	75%	4312.000000	3603.250000	4169.750000	4247.250000
	max	4732.000000	5140.000000	4633.000000	4997.000000
acceleration	count	29.000000	28.000000	28.000000	40.000000
	mean	12.948276	15.142857	15.125000	14.312500
	std	3.330982	2.666171	2.850032	2.754222
	min	8.000000	11.500000	11.000000	9.500000
	25%	10.000000	13.375000	13.375000	12.500000
	50%	12.500000	14.500000	14.500000	14.000000
	75%	15.000000	16.125000	16.625000	16.000000
	max	20.500000	20.500000	23.500000	21.000000
origin	count	29.000000	28.000000	28.000000	40.000000
	mean	1.310345	1.428571	1.535714	1.375000
	std	0.603765	0.741798	0.792658	0.667467
	min	1.000000	1.000000	1.000000	1.000000
	25%	1.000000	1.000000	1.000000	1.000000
	50%	1.000000	1.000000	1.000000	1.000000
	75%	1.000000	2.000000	2.000000	2.000000
	max	3.000000	3.000000	3.000000	3.000000

model_year		74	75	76	77
\					
mpg	count	27.000000	30.000000	34.000000	28.000000
	mean	22.703704	20.266667	21.573529	23.375000
	std	6.420010	4.940566	5.889297	6.675862
	min	13.000000	13.000000	13.000000	15.000000
	25%	16.000000	16.000000	16.750000	17.375000
	50%	24.000000	19.500000	21.000000	21.750000
	75%	27.000000	23.000000	26.375000	30.000000

	max	32.000000	33.000000	33.000000	36.000000
cylinders	count	27.000000	30.000000	34.000000	28.000000
	mean	5.259259	5.600000	5.647059	5.464286
	std	1.583390	1.522249	1.667558	1.815206
	min	4.000000	4.000000	4.000000	3.000000
	25%	4.000000	4.000000	4.000000	4.000000
	50%	4.000000	6.000000	6.000000	4.000000
	75%	6.000000	6.000000	7.500000	8.000000
	max	8.000000	8.000000	8.000000	8.000000
displacement	count	27.000000	30.000000	34.000000	28.000000
	mean	171.740741	205.533333	197.794118	191.392857
	std	92.601127	87.669730	94.422256	107.813742
	min	71.000000	90.000000	85.000000	79.000000
	25%	90.000000	121.000000	102.500000	97.750000
	50%	122.000000	228.000000	184.000000	143.000000
	75%	250.000000	250.000000	291.000000	270.500000
	max	350.000000	400.000000	351.000000	400.000000
weight	count	27.000000	30.000000	34.000000	28.000000
	mean	2877.925926	3176.800000	3078.735294	2997.357143
	std	949.308571	765.179781	821.371481	912.825902
	min	1649.000000	1795.000000	1795.000000	1825.000000
	25%	2116.500000	2676.750000	2228.750000	2135.000000
	50%	2489.000000	3098.500000	3171.500000	2747.500000
	75%	3622.500000	3662.250000	3803.750000	3925.000000
	max	4699.000000	4668.000000	4380.000000	4335.000000
acceleration	count	27.000000	30.000000	34.000000	28.000000

```
              mean     16.203704    16.050000    15.941176    15.435714
              std       1.688532     2.471737     2.801419     2.273391
              min      13.500000    11.500000    12.000000    11.100000
              25%      15.250000    14.125000    13.925000    14.000000
              50%      16.000000    16.000000    15.500000    15.650000
              75%      17.000000    17.375000    17.550000    16.925000
              max      21.000000    21.000000    22.200000    19.000000
origin        count    27.000000    30.000000    34.000000    28.000000
              mean      1.666667     1.466667     1.470588     1.571429
              std       0.832050     0.730297     0.706476     0.835711
              min       1.000000     1.000000     1.000000     1.000000
              25%       1.000000     1.000000     1.000000     1.000000
              50%       1.000000     1.000000     1.000000     1.000000
              75%       2.000000     2.000000     2.000000     2.000000
              max       3.000000     3.000000     3.000000     3.000000

model_year                  78           79           80           81
\
mpg           count    36.000000    29.000000    29.000000    29.000000
              mean     24.061111    25.093103    33.696552    30.334483
              std       6.898044     6.794217     7.037983     5.591465
              min      16.200000    15.500000    19.100000    17.600000
              25%      19.350000    19.200000    29.800000    26.600000
              50%      20.700000    23.900000    32.700000    31.600000
              75%      28.000000    31.800000    38.100000    34.400000
              max      43.100000    37.300000    46.600000    39.100000
cylinders     count    36.000000    29.000000    29.000000    29.000000
```

	mean	5.361111	5.827586	4.137931	4.620690
	std	1.495761	1.774199	0.580895	1.082781
	min	4.000000	4.000000	3.000000	4.000000
	25%	4.000000	4.000000	4.000000	4.000000
	50%	5.500000	6.000000	4.000000	4.000000
	75%	6.000000	8.000000	4.000000	6.000000
	max	8.000000	8.000000	6.000000	8.000000
displacement	count	36.000000	29.000000	29.000000	29.000000
	mean	177.805556	206.689655	115.827586	135.310345
	std	76.012713	96.307581	33.744914	58.387929
	min	78.000000	85.000000	70.000000	79.000000
	25%	115.500000	121.000000	90.000000	98.000000
	50%	159.500000	183.000000	107.000000	119.000000
	75%	231.000000	302.000000	140.000000	151.000000
	max	318.000000	360.000000	225.000000	350.000000
weight	count	36.000000	29.000000	29.000000	29.000000
	mean	2861.805556	3055.344828	2436.655172	2522.931034
	std	626.023907	747.881497	432.235491	533.600501
	min	1800.000000	1915.000000	1835.000000	1755.000000
	25%	2282.500000	2556.000000	2110.000000	2065.000000
	50%	2910.000000	3190.000000	2335.000000	2385.000000
	75%	3410.000000	3725.000000	2800.000000	2900.000000
	max	4080.000000	4360.000000	3381.000000	3725.000000
acceleration	count	36.000000	29.000000	29.000000	29.000000
	mean	15.805556	15.813793	16.934483	16.306897
	std	2.129915	2.952931	2.826694	2.192509

```
             min      11.200000    11.300000    11.400000    12.600000
             25%      14.475000    14.000000    15.100000    14.800000
             50%      15.750000    15.000000    16.500000    16.200000
             75%      16.825000    17.300000    18.700000    17.300000
             max      21.500000    24.800000    23.700000    20.700000
origin       count    36.000000    29.000000    29.000000    29.000000
             mean      1.611111     1.275862     2.206897     1.965517
             std       0.837608     0.591400     0.818505     0.944259
             min       1.000000     1.000000     1.000000     1.000000
             25%       1.000000     1.000000     2.000000     1.000000
             50%       1.000000     1.000000     2.000000     2.000000
             75%       2.000000     1.000000     3.000000     3.000000
             max       3.000000     3.000000     3.000000     3.000000


model_year                  82
mpg          count    31.000000
             mean     31.709677
             std       5.392548
             min      22.000000
             25%      27.000000
             50%      32.000000
             75%      36.000000
             max      44.000000
cylinders    count    31.000000
             mean      4.193548
             std       0.601074
             min       4.000000
             25%       4.000000
             50%       4.000000
             75%       4.000000
             max       6.000000
displacement count    31.000000
             mean    128.870968
             std      39.352037
             min      91.000000
             25%     105.000000
             50%     119.000000
             75%     142.000000
```

```
              max     262.000000
weight        count    31.000000
              mean   2453.548387
              std     354.276713
              min    1965.000000
              25%    2127.500000
              50%    2525.000000
              75%    2727.500000
              max    3035.000000
acceleration  count    31.000000
              mean     16.638710
              std       2.484844
              min      11.600000
              25%      14.850000
              50%      16.400000
              75%      18.000000
              max      24.600000
origin        count    31.000000
              mean      1.645161
              std       0.914636
              min       1.000000
              25%       1.000000
              50%       1.000000
              75%       3.000000
              max       3.000000
```

```
# Questa tabella è super leggibile!!!

# RIPARTIAMO DAL NOSTRO Df CON RAGGRUPPAMENTO:

a = mpg.groupby(['model_year','origin'])
[['cylinders','weight','acceleration']].mean()
a
```

		cylinders	weight	acceleration
model_year	origin			
70	1	7.636364	3716.500000	11.977273
	2	4.000000	2309.200000	16.500000
	3	4.000000	2251.000000	14.750000
71	1	6.200000	3401.600000	14.575000
	2	4.000000	2024.000000	16.750000
	3	4.000000	1936.000000	16.375000
72	1	6.888889	3682.666667	14.055556
	2	4.000000	2573.200000	18.700000
	3	3.800000	2300.400000	15.400000
73	1	7.241379	3821.448276	13.620690
	2	4.000000	2335.714286	16.428571
	3	4.250000	2397.250000	15.625000

74	1	6.266667	3503.333333	15.966667
	2	4.000000	2139.333333	15.333333
	3	4.000000	2053.000000	17.666667
75	1	6.400000	3533.200000	16.350000
	2	4.000000	2571.166667	15.083333
	3	4.000000	2303.250000	16.000000
76	1	6.363636	3405.409091	15.786364
	2	4.250000	2611.000000	16.050000
	3	4.500000	2217.500000	16.575000
77	1	6.222222	3422.000000	15.238889
	2	4.000000	2138.750000	15.000000
	3	4.166667	2295.833333	16.316667
78	1	6.000000	3141.136364	15.545455
	2	4.833333	2691.666667	16.233333
	3	4.000000	2221.250000	16.200000
79	1	6.260870	3210.217391	15.243478
	2	4.250000	2693.750000	18.400000
	3	4.000000	1997.500000	17.200000
80	1	4.285714	2822.428571	16.800000
	2	4.111111	2348.000000	18.366667
	3	4.076923	2290.307692	16.015385
81	1	4.923077	2695.000000	16.053846
	2	4.500000	2725.000000	17.500000
	3	4.333333	2269.166667	16.183333
82	1	4.300000	2637.750000	16.670000
	2	4.000000	2055.000000	19.950000
	3	4.000000	2132.777778	15.833333

```
# ESTRAIAMO QUINDI IL NOME DElle COLONNE INDICE:

a.index.names
```

```
FrozenList(['model_year', 'origin'])
```

```
# POSSIAMO ANCHE ESTRARRE I VALORI CHE GLI INDICI POSSONO ASSUMERE:

a.index.levels
```

```
FrozenList([[70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82], [1,
2, 3]])
```

```
# La prima lista indica i valori che la variabile categorica più
esterna assume. La seconda riguarda la variabile categorica pi
interna.

# RICORDA: SIA LA VARIABILE PIU ESTERNA SIA QUELLA PIU INTERNA DEVONO
ESSERE CATEGORICHE!!!
```

```
# A partire dalla variabile più esterna assunta come indice, possiamo estrarre l'indice più interno ed i valori corrispondenti tramite la funzione loc
```

```
a.loc[70]
```

```
        cylinders  weight  acceleration
origin
1        7.636364  3716.5     11.977273
2        4.000000  2309.2     16.500000
3        4.000000  2251.0     14.750000
```

```
# Possiamo anche ragionare con più anni
```

```
a.loc[[70,75,81]]
```

```
                   cylinders       weight  acceleration
model_year origin
70         1        7.636364  3716.500000     11.977273
           2        4.000000  2309.200000     16.500000
           3        4.000000  2251.000000     14.750000
75         1        6.400000  3533.200000     16.350000
           2        4.000000  2571.166667     15.083333
           3        4.000000  2303.250000     16.000000
81         1        4.923077  2695.000000     16.053846
           2        4.500000  2725.000000     17.500000
           3        4.333333  2269.166667     16.183333
```

```
# Possiamo anche passare una tupla contenente il valore dell'indice più esterno e di quello più interno. In questo modo
# otterremo una Series contenente i valori corrispondenti. Ripartendo da a:
```

```
a
```

```
                   cylinders       weight  acceleration
model_year origin
70         1        7.636364  3716.500000     11.977273
           2        4.000000  2309.200000     16.500000
           3        4.000000  2251.000000     14.750000
71         1        6.200000  3401.600000     14.575000
           2        4.000000  2024.000000     16.750000
           3        4.000000  1936.000000     16.375000
72         1        6.888889  3682.666667     14.055556
```

```
           2         4.000000  2573.200000     18.700000
           3         3.800000  2300.400000     15.400000
73         1         7.241379  3821.448276     13.620690
           2         4.000000  2335.714286     16.428571
           3         4.250000  2397.250000     15.625000
74         1         6.266667  3503.333333     15.966667
           2         4.000000  2139.333333     15.333333
           3         4.000000  2053.000000     17.666667
75         1         6.400000  3533.200000     16.350000
           2         4.000000  2571.166667     15.083333
           3         4.000000  2303.250000     16.000000
76         1         6.363636  3405.409091     15.786364
           2         4.250000  2611.000000     16.050000
           3         4.500000  2217.500000     16.575000
77         1         6.222222  3422.000000     15.238889
           2         4.000000  2138.750000     15.000000
           3         4.166667  2295.833333     16.316667
78         1         6.000000  3141.136364     15.545455
           2         4.833333  2691.666667     16.233333
           3         4.000000  2221.250000     16.200000
79         1         6.260870  3210.217391     15.243478
           2         4.250000  2693.750000     18.400000
           3         4.000000  1997.500000     17.200000
80         1         4.285714  2822.428571     16.800000
           2         4.111111  2348.000000     18.366667
           3         4.076923  2290.307692     16.015385
81         1         4.923077  2695.000000     16.053846
           2         4.500000  2725.000000     17.500000
           3         4.333333  2269.166667     16.183333
82         1         4.300000  2637.750000     16.670000
           2         4.000000  2055.000000     19.950000
           3         4.000000  2132.777778     15.833333
```

```
a.loc[(79,3)]
```

```
cylinders          4.0
weight          1997.5
acceleration      17.2
Name: (79, 3), dtype: float64
```

```
# CVD


# Supponiamo ora di voler estrarre solo l'indice più interno
desiderato. Ad esempio, supponiamo di voler avere
# i dati relativi ai valori di origin 1,2,3 relativi al model_year
pari sa 80. Per farlo, si utilizza
# la funzione xs (Cross Section):
```

```
a.xs(key=80, level = "model_year")
```

```
        cylinders       weight  acceleration
origin
1        4.285714  2822.428571     16.800000
2        4.111111  2348.000000     18.366667
3        4.076923  2290.307692     16.015385
```

```
# CVD
```

```
# Supponeamo di ovler estrarre più di un level. Supponiamo, ad es, di voler estrarre i level 89,81,82 e i dati relativi.
# In questo caso non possiamo utilizzare xs, ma loc:
```

```
a.loc[[80,81,82]]
```

```
                    cylinders       weight  acceleration
model_year origin
80         1         4.285714  2822.428571     16.800000
           2         4.111111  2348.000000     18.366667
           3         4.076923  2290.307692     16.015385
81         1         4.923077  2695.000000     16.053846
           2         4.500000  2725.000000     17.500000
           3         4.333333  2269.166667     16.183333
82         1         4.300000  2637.750000     16.670000
           2         4.000000  2055.000000     19.950000
           3         4.000000  2132.777778     15.833333
```

```
# Torniamo alla nostra tabella raggruppata:
```

```
a
```

```
                    cylinders       weight  acceleration
model_year origin
70         1         7.636364  3716.500000     11.977273
           2         4.000000  2309.200000     16.500000
           3         4.000000  2251.000000     14.750000
71         1         6.200000  3401.600000     14.575000
           2         4.000000  2024.000000     16.750000
           3         4.000000  1936.000000     16.375000
72         1         6.888889  3682.666667     14.055556
```

```
            2         4.000000  2573.200000     18.700000
            3         3.800000  2300.400000     15.400000
73          1         7.241379  3821.448276     13.620690
            2         4.000000  2335.714286     16.428571
            3         4.250000  2397.250000     15.625000
74          1         6.266667  3503.333333     15.966667
            2         4.000000  2139.333333     15.333333
            3         4.000000  2053.000000     17.666667
75          1         6.400000  3533.200000     16.350000
            2         4.000000  2571.166667     15.083333
            3         4.000000  2303.250000     16.000000
76          1         6.363636  3405.409091     15.786364
            2         4.250000  2611.000000     16.050000
            3         4.500000  2217.500000     16.575000
77          1         6.222222  3422.000000     15.238889
            2         4.000000  2138.750000     15.000000
            3         4.166667  2295.833333     16.316667
78          1         6.000000  3141.136364     15.545455
            2         4.833333  2691.666667     16.233333
            3         4.000000  2221.250000     16.200000
79          1         6.260870  3210.217391     15.243478
            2         4.250000  2693.750000     18.400000
            3         4.000000  1997.500000     17.200000
80          1         4.285714  2822.428571     16.800000
            2         4.111111  2348.000000     18.366667
            3         4.076923  2290.307692     16.015385
81          1         4.923077  2695.000000     16.053846
            2         4.500000  2725.000000     17.500000
            3         4.333333  2269.166667     16.183333
82          1         4.300000  2637.750000     16.670000
            2         4.000000  2055.000000     19.950000
            3         4.000000  2132.777778     15.833333
```

```
# Supponiamo di voler estrarre i valori 3 relativi a origin PER TUTTI
GLI ANNI.
# Per farlo, possiamo utilizzare sempre la funzione xs,
# ma il parametro level deve essere valorizzato con origin
```

```
a.xs(key=3, level="origin")
```

```
            cylinders       weight  acceleration
model_year
70           4.000000  2251.000000     14.750000
71           4.000000  1936.000000     16.375000
72           3.800000  2300.400000     15.400000
73           4.250000  2397.250000     15.625000
74           4.000000  2053.000000     17.666667
```

```
75           4.000000  2303.250000     16.000000
76           4.500000  2217.500000     16.575000
77           4.166667  2295.833333     16.316667
78           4.000000  2221.250000     16.200000
79           4.000000  1997.500000     17.200000
80           4.076923  2290.307692     16.015385
81           4.333333  2269.166667     16.183333
82           4.000000  2132.777778     15.833333
```

```
# Ovviamente la colonna origin è stata eliminata
```

```
a
```

```
                  cylinders       weight  acceleration
model_year origin
70         1       7.636364  3716.500000     11.977273
           2       4.000000  2309.200000     16.500000
           3       4.000000  2251.000000     14.750000
71         1       6.200000  3401.600000     14.575000
           2       4.000000  2024.000000     16.750000
           3       4.000000  1936.000000     16.375000
72         1       6.888889  3682.666667     14.055556
           2       4.000000  2573.200000     18.700000
           3       3.800000  2300.400000     15.400000
73         1       7.241379  3821.448276     13.620690
           2       4.000000  2335.714286     16.428571
           3       4.250000  2397.250000     15.625000
74         1       6.266667  3503.333333     15.966667
           2       4.000000  2139.333333     15.333333
           3       4.000000  2053.000000     17.666667
75         1       6.400000  3533.200000     16.350000
           2       4.000000  2571.166667     15.083333
           3       4.000000  2303.250000     16.000000
76         1       6.363636  3405.409091     15.786364
           2       4.250000  2611.000000     16.050000
           3       4.500000  2217.500000     16.575000
77         1       6.222222  3422.000000     15.238889
           2       4.000000  2138.750000     15.000000
           3       4.166667  2295.833333     16.316667
78         1       6.000000  3141.136364     15.545455
           2       4.833333  2691.666667     16.233333
           3       4.000000  2221.250000     16.200000
79         1       6.260870  3210.217391     15.243478
           2       4.250000  2693.750000     18.400000
           3       4.000000  1997.500000     17.200000
80         1       4.285714  2822.428571     16.800000
           2       4.111111  2348.000000     18.366667
```

```
           3        4.076923  2290.307692     16.015385
81         1        4.923077  2695.000000     16.053846
           2        4.500000  2725.000000     17.500000
           3        4.333333  2269.166667     16.183333
82         1        4.300000  2637.750000     16.670000
           2        4.000000  2055.000000     19.950000
           3        4.000000  2132.777778     15.833333
```

```
# E se volessimo estrarre più di un valore di origin PER TUTTI GLI
ANNI?
# Supponiamo, ad esempio, di voler estrarre i dati relativi ai valori
# 2 e 3 della colonna origin. Non conviene utilizzare xs.
# Per prima cosa torniamo al dataframe di partenza:
```

```
mpg
```

```
      mpg  cylinders  displacement horsepower  weight  acceleration  \
0    18.0          8         307.0        130    3504          12.0
1    15.0          8         350.0        165    3693          11.5
2    18.0          8         318.0        150    3436          11.0
3    16.0          8         304.0        150    3433          12.0
4    17.0          8         302.0        140    3449          10.5
..    ...        ...           ...        ...     ...           ...
393  27.0          4         140.0         86    2790          15.6
394  44.0          4          97.0         52    2130          24.6
395  32.0          4         135.0         84    2295          11.6
396  28.0          4         120.0         79    2625          18.6
397  31.0          4         119.0         82    2720          19.4

     model_year  origin                       name
0            70       1  chevrolet chevelle malibu
1            70       1          buick skylark 320
2            70       1         plymouth satellite
3            70       1              amc rebel sst
4            70       1                ford torino
..          ...     ...                        ...
393          82       1            ford mustang gl
394          82       2                  vw pickup
395          82       1              dodge rampage
396          82       1                ford ranger
397          82       1                 chevy s-10

[398 rows x 9 columns]
```

```
# A questo punto filtriamo il Df per i valori 1 e 3 della colonna
origin:
```

```
mask = (mpg["origin"] == 1) | (mpg["origin"] == 3)   # Oppure mpg["origin"].isin([1,3])
mpg = mpg[mask]
mpg
```

```
      mpg  cylinders  displacement horsepower  weight  acceleration  \
0    18.0          8         307.0        130    3504          12.0
1    15.0          8         350.0        165    3693          11.5
2    18.0          8         318.0        150    3436          11.0
3    16.0          8         304.0        150    3433          12.0
4    17.0          8         302.0        140    3449          10.5
..    ...        ...           ...        ...     ...           ...
392  27.0          4         151.0         90    2950          17.3
393  27.0          4         140.0         86    2790          15.6
395  32.0          4         135.0         84    2295          11.6
396  28.0          4         120.0         79    2625          18.6
397  31.0          4         119.0         82    2720          19.4

     model_year  origin                       name
0            70       1  chevrolet chevelle malibu
1            70       1          buick skylark 320
2            70       1         plymouth satellite
3            70       1              amc rebel sst
4            70       1                ford torino
..          ...     ...                        ...
392          82       1           chevrolet camaro
393          82       1            ford mustang gl
395          82       1              dodge rampage
396          82       1                ford ranger
397          82       1                 chevy s-10

[328 rows x 9 columns]
```

```
mpg["origin"].value_counts()
```

```
origin
1    249
3     79
Name: count, dtype: int64
```

```
# CVD.
```

```
# A questo punto possiamo fare la group by relativa a model_year ed a origin
```

```
mpg_grouped  = mpg.groupby(["model_year","origin"])
[mpg.select_dtypes(include='number').columns].sum()
mpg_grouped
```

```
                      mpg  cylinders  displacement  weight
acceleration  \
model_year origin

70         1        336.0        168        7412.0   81763
263.5
           3         51.0          8         210.0    4502
29.5
71         1        362.0        124        5140.0   68032
291.5
           3        118.0         16         353.0    7744
65.5
72         1        293.0        124        5062.5   66288
253.0
           3        121.0         19         497.0   11502
77.0
73         1        436.0        210        9109.0  110822
395.0
           3         80.0         17         431.0    9589
62.5
74         1        275.0         94        3541.0   52550
239.5
           3        176.0         24         537.0   12318
106.0
75         1        351.0        128        5068.0   70664
327.0
           3        110.0         16         441.0    9213
64.0
76         1        427.5        140        5367.0   74919
347.3
           3        112.0         18         429.0    8870
66.3
77         1        373.0        112        4362.0   61596
274.3
           3        164.5         25         603.0   13775
97.9
78         1        479.0        132        4786.0   69105
342.0
           3        237.5         32         858.0   17770
129.6
79         1        540.0        144        5319.0   73835
350.6
           3         65.9          8         171.0    3995
34.4
80         1        181.4         30        1061.0   19757
117.6
```

```
            3       460.2          53        1360.0   29774
208.2
81          1       357.9          64        2143.0   35035
208.7
            3       395.5          52        1290.0   27230
194.2
82          1       589.0          86        2859.0   52755
333.4
            3       314.0          36         934.0   19195
142.5

                    model_year  origin
model_year origin
70          1             1540      22
            3              140       6
71          1             1420      20
            3              284      12
72          1             1296      18
            3              360      15
73          1             2117      29
            3              292      12
74          1             1110      15
            3              444      18
75          1             1500      20
            3              300      12
76          1             1672      22
            3              304      12
77          1             1386      18
            3              462      18
78          1             1716      22
            3              624      24
79          1             1817      23
            3              158       6
80          1              560       7
            3             1040      39
81          1             1053      13
            3              972      36
82          1             1640      20
            3              738      27


# Ed ecco he i valori relativi ad origin sono 1 e 3, CVD



# Possiamo anche invertire i nostri level. Possiamo qundi invertire
# model_year e origin tramite swaplevel
```

```
mpg_grouped.swaplevel()
```

```
                     mpg  cylinders  displacement  weight
acceleration  \
origin model_year

1      70          336.0        168        7412.0   81763
263.5
3      70           51.0          8         210.0    4502
29.5
1      71          362.0        124        5140.0   68032
291.5
3      71          118.0         16         353.0    7744
65.5
1      72          293.0        124        5062.5   66288
253.0
3      72          121.0         19         497.0   11502
77.0
1      73          436.0        210        9109.0  110822
395.0
3      73           80.0         17         431.0    9589
62.5
1      74          275.0         94        3541.0   52550
239.5
3      74          176.0         24         537.0   12318
106.0
1      75          351.0        128        5068.0   70664
327.0
3      75          110.0         16         441.0    9213
64.0
1      76          427.5        140        5367.0   74919
347.3
3      76          112.0         18         429.0    8870
66.3
1      77          373.0        112        4362.0   61596
274.3
3      77          164.5         25         603.0   13775
97.9
1      78          479.0        132        4786.0   69105
342.0
3      78          237.5         32         858.0   17770
129.6
1      79          540.0        144        5319.0   73835
350.6
3      79           65.9          8         171.0    3995
34.4
1      80          181.4         30        1061.0   19757
117.6
```

```
3      80           460.2          53         1360.0   29774
208.2
1      81           357.9          64         2143.0   35035
208.7
3      81           395.5          52         1290.0   27230
194.2
1      82           589.0          86         2859.0   52755
333.4
3      82           314.0          36          934.0   19195
142.5

                    model_year  origin
origin model_year
1      70                 1540      22
3      70                  140       6
1      71                 1420      20
3      71                  284      12
1      72                 1296      18
3      72                  360      15
1      73                 2117      29
3      73                  292      12
1      74                 1110      15
3      74                  444      18
1      75                 1500      20
3      75                  300      12
1      76                 1672      22
3      76                  304      12
1      77                 1386      18
3      77                  462      18
1      78                 1716      22
3      78                  624      24
1      79                 1817      23
3      79                  158       6
1      80                  560       7
3      80                 1040      39
1      81                 1053      13
3      81                  972      36
1      82                 1640      20
3      82                  738      27



mpg_grouped

                      mpg  cylinders  displacement  weight
acceleration  \
model_year origin

70         1       336.0        168        7412.0   81763
263.5
```

```
            3         51.0           8          210.0    4502
29.5
71          1        362.0         124         5140.0   68032
291.5
            3        118.0          16          353.0    7744
65.5
72          1        293.0         124         5062.5   66288
253.0
            3        121.0          19          497.0   11502
77.0
73          1        436.0         210         9109.0  110822
395.0
            3         80.0          17          431.0    9589
62.5
74          1        275.0          94         3541.0   52550
239.5
            3        176.0          24          537.0   12318
106.0
75          1        351.0         128         5068.0   70664
327.0
            3        110.0          16          441.0    9213
64.0
76          1        427.5         140         5367.0   74919
347.3
            3        112.0          18          429.0    8870
66.3
77          1        373.0         112         4362.0   61596
274.3
            3        164.5          25          603.0   13775
97.9
78          1        479.0         132         4786.0   69105
342.0
            3        237.5          32          858.0   17770
129.6
79          1        540.0         144         5319.0   73835
350.6
            3         65.9           8          171.0    3995
34.4
80          1        181.4          30         1061.0   19757
117.6
            3        460.2          53         1360.0   29774
208.2
81          1        357.9          64         2143.0   35035
208.7
            3        395.5          52         1290.0   27230
194.2
82          1        589.0          86         2859.0   52755
333.4
            3        314.0          36          934.0   19195
```

```
142.5

                    model_year  origin
model_year origin
70         1              1540      22
           3               140       6
71         1              1420      20
           3               284      12
72         1              1296      18
           3               360      15
73         1              2117      29
           3               292      12
74         1              1110      15
           3               444      18
75         1              1500      20
           3               300      12
76         1              1672      22
           3               304      12
77         1              1386      18
           3               462      18
78         1              1716      22
           3               624      24
79         1              1817      23
           3               158       6
80         1               560       7
           3              1040      39
81         1              1053      13
           3               972      36
82         1              1640      20
           3               738      27


# Possiamo anche ordinare i valori di model_year in ordine decrescente


mpg_grouped.sort_index(level = "model_year", ascending = False )

                     mpg  cylinders  displacement  weight
acceleration  \
model_year origin

82         3       314.0         36         934.0   19195
142.5
           1       589.0         86        2859.0   52755
333.4
81         3       395.5         52        1290.0   27230
194.2
           1       357.9         64        2143.0   35035
208.7
```

```
80          3       460.2          53        1360.0   29774
208.2
            1       181.4          30        1061.0   19757
117.6
79          3        65.9           8         171.0    3995
34.4
            1       540.0         144        5319.0   73835
350.6
78          3       237.5          32         858.0   17770
129.6
            1       479.0         132        4786.0   69105
342.0
77          3       164.5          25         603.0   13775
97.9
            1       373.0         112        4362.0   61596
274.3
76          3       112.0          18         429.0    8870
66.3
            1       427.5         140        5367.0   74919
347.3
75          3       110.0          16         441.0    9213
64.0
            1       351.0         128        5068.0   70664
327.0
74          3       176.0          24         537.0   12318
106.0
            1       275.0          94        3541.0   52550
239.5
73          3        80.0          17         431.0    9589
62.5
            1       436.0         210        9109.0  110822
395.0
72          3       121.0          19         497.0   11502
77.0
            1       293.0         124        5062.5   66288
253.0
71          3       118.0          16         353.0    7744
65.5
            1       362.0         124        5140.0   68032
291.5
70          3        51.0           8         210.0    4502
29.5
            1       336.0         168        7412.0   81763
263.5

                  model_year  origin
model_year origin
82          3            738      27
            1           1640      20
```

```
81          3               972       36
            1              1053       13
80          3              1040       39
            1               560        7
79          3               158        6
            1              1817       23
78          3               624       24
            1              1716       22
77          3               462       18
            1              1386       18
76          3               304       12
            1              1672       22
75          3               300       12
            1              1500       20
74          3               444       18
            1              1110       15
73          3               292       12
            1              2117       29
72          3               360       15
            1              1296       18
71          3               284       12
            1              1420       20
70          3               140        6
            1              1540       22
```

```
# Se invece effettuassimo l'ordinamento per origin, otterremo dei
duplicati nella colonna model_year
```

```
mpg_grouped.sort_index(level = "origin", ascending = False )
```

```
                     mpg  cylinders  displacement  weight
acceleration  \
model_year origin

82         3       314.0         36         934.0   19195
142.5
81         3       395.5         52        1290.0   27230
194.2
80         3       460.2         53        1360.0   29774
208.2
79         3        65.9          8         171.0    3995
34.4
78         3       237.5         32         858.0   17770
129.6
77         3       164.5         25         603.0   13775
97.9
76         3       112.0         18         429.0    8870
```

```
66.3
75          3       110.0          16          441.0    9213
64.0
74          3       176.0          24          537.0   12318
106.0
73          3        80.0          17          431.0    9589
62.5
72          3       121.0          19          497.0   11502
77.0
71          3       118.0          16          353.0    7744
65.5
70          3        51.0           8          210.0    4502
29.5
82          1       589.0          86         2859.0   52755
333.4
81          1       357.9          64         2143.0   35035
208.7
80          1       181.4          30         1061.0   19757
117.6
79          1       540.0         144         5319.0   73835
350.6
78          1       479.0         132         4786.0   69105
342.0
77          1       373.0         112         4362.0   61596
274.3
76          1       427.5         140         5367.0   74919
347.3
75          1       351.0         128         5068.0   70664
327.0
74          1       275.0          94         3541.0   52550
239.5
73          1       436.0         210         9109.0  110822
395.0
72          1       293.0         124         5062.5   66288
253.0
71          1       362.0         124         5140.0   68032
291.5
70          1       336.0         168         7412.0   81763
263.5

                  model_year  origin
model_year origin
82          3              738      27
81          3              972      36
80          3             1040      39
79          3              158       6
78          3              624      24
77          3              462      18
76          3              304      12
```

```
75          3               300      12
74          3               444      18
73          3               292      12
72          3               360      15
71          3               284      12
70          3               140       6
82          1              1640      20
81          1              1053      13
80          1               560       7
79          1              1817      23
78          1              1716      22
77          1              1386      18
76          1              1672      22
75          1              1500      20
74          1              1110      15
73          1              2117      29
72          1              1296      18
71          1              1420      20
70          1              1540      22
```

```
# CVD

# Torniamo al nostro df iniziale

mpg.head()
```

```
    mpg  cylinders  displacement horsepower  weight  acceleration
model_year  \
0  18.0          8         307.0        130    3504          12.0
70
1  15.0          8         350.0        165    3693          11.5
70
2  18.0          8         318.0        150    3436          11.0
70
3  16.0          8         304.0        150    3433          12.0
70
4  17.0          8         302.0        140    3449          10.5
70

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

```
# Statistiche di base, metodo ARG-

# Finora abbiamo calcolato le statistiche di base per singola colonna.
# Possiamo calcolare più velocemente le statistiche di base tramite la
funzione AGG.
# Per prima cosa, però, dobbiamo estrarre dal nostro Df SOLO le
colonne numeriche


df_numeriche = mpg.select_dtypes(include=['number'])
df_numeriche
```

```
      mpg  cylinders  displacement  weight  acceleration  model_year
origin
0    18.0          8         307.0    3504          12.0          70
1
1    15.0          8         350.0    3693          11.5          70
1
2    18.0          8         318.0    3436          11.0          70
1
3    16.0          8         304.0    3433          12.0          70
1
4    17.0          8         302.0    3449          10.5          70
1
..    ...        ...           ...     ...           ...         ...
...
392  27.0          4         151.0    2950          17.3          82
1
393  27.0          4         140.0    2790          15.6          82
1
395  32.0          4         135.0    2295          11.6          82
1
396  28.0          4         120.0    2625          18.6          82
1
397  31.0          4         119.0    2720          19.4          82
1

[328 rows x 7 columns]
```

```
df_numeriche.agg(["std","mean"]).transpose()
```

```
                     std         mean
mpg             7.723398    22.580488
cylinders       1.738563     5.731707
displacement  106.050898   211.413110
weight        861.527429  3087.189024
acceleration    2.624770    15.307927
```

```
model_year      3.748239    76.051829
origin          0.856510     1.481707
```

```
# Ovviamente ha poco senso effettuare media e ev std per alcune di
queste colonne. Il tutto è solo a titolo di esempio.
# Se volessimo selezionare solo alcune colonne, tipo weight e
acceleration, possiamo filtrare il Df
# precedente per queste due variabili
```

```
df_numeriche.agg(["std","mean"])
[["weight","acceleration"]].transpose()
```

```
                     std         mean
weight        861.527429  3087.189024
acceleration    2.624770    15.307927
```

```
# Oppure, possiamo passare un dizionario nella nostra funzione agg
```

```
df_numeriche.agg({"weight":["std","mean"],"acceleration":
["std","mean"]}).transpose()
```

```
                     std         mean
weight        861.527429  3087.189024
acceleration    2.624770    15.307927
```