# "Cirugía de datos"

Es el proceso de transformar y mapear datos de un dataset raw (en bruto) en otro formato con la intención de hacerlo más apropiado y valioso para una variedad de propósitos posteriores, como el análisis.

Este proceso puede incluir visualización de datos, agregación de datos, entrenamiento de un modelo estadístico, así como muchos otros usos potenciales.

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
data = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/customer-churn-model/Customer Churn Model.txt')
```

In [3]:
```
data.head()
```

Out[3]:

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Calls | Day Charge | ... | Eve Calls | Eve Charge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KS | 128 | 415 | 382-4657 | no | yes | 25 | 265.1 | 110 | 45.07 | ... | 99 | 16.78 |
| **1** | OH | 107 | 415 | 371-7191 | no | yes | 26 | 161.6 | 123 | 27.47 | ... | 103 | 16.62 |
| **2** | NJ | 137 | 415 | 358-1921 | no | no | 0 | 243.4 | 114 | 41.38 | ... | 110 | 10.30 |
| **3** | OH | 84 | 408 | 375-9999 | yes | no | 0 | 299.4 | 71 | 50.90 | ... | 88 | 5.26 |
| **4** | OK | 75 | 415 | 330-6626 | yes | no | 0 | 166.7 | 113 | 28.34 | ... | 122 | 12.61 |

5 rows × 21 columns

## Seleccionar "n" columnas para un nuevo subset

In [4]:
```
data.columns
```

Out[4]:
```
Index(['State', 'Account Length', 'Area Code', 'Phone', 'Int'l Plan',
       'VMail Plan', 'VMail Message', 'Day Mins', 'Day Calls', 'Day Charge',
       'Eve Mins', 'Eve Calls', 'Eve Charge', 'Night Mins', 'Night Calls',
       'Night Charge', 'Intl Mins', 'Intl Calls', 'Intl Charge',
       'CustServ Calls', 'Churn?'],
      dtype='object')
```

In [5]:
```
columnas_deseadas = ['Account Length', 'Phone', 'Eve Charge']
subset = data[columnas_deseadas]
subset.head()
```

Out[5]:

| | Account Length | Phone | Eve Charge |
|---|---|---|---|
| **0** | 128 | 382-4657 | 16.78 |
| **1** | 107 | 371-7191 | 16.62 |
| **2** | 137 | 358-1921 | 10.30 |
| **3** | 84 | 375-9999 | 5.26 |
| **4** | 75 | 330-6626 | 12.61 |

In [6]:
```
#Obtener las columnas deseadas, cuando las que no se quieren son pocas
columnas_no_deseadas = ['Account Leght', 'Vmail Message', 'Day Calls']
all_columnas = data.columns.values.tolist()

#Obtener todas las columnas que se quieren
sublist = [x for x in all_columnas if x not in columnas_no_deseadas]

subset2 = data[sublist]
subset2
```

Out[6]:

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Charge | Eve Mins | Eve Calls | Eve Charge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KS | 128 | 415 | 382-4657 | no | yes | 25 | 265.1 | 45.07 | 197.4 | 99 | 16.78 |
| **1** | OH | 107 | 415 | 371-7191 | no | yes | 26 | 161.6 | 27.47 | 195.5 | 103 | 16.62 |
| **2** | NJ | 137 | 415 | 358-1921 | no | no | 0 | 243.4 | 41.38 | 121.2 | 110 | 10.30 |
| **3** | OH | 84 | 408 | 375-9999 | yes | no | 0 | 299.4 | 50.90 | 61.9 | 88 | 5.26 |
| **4** | OK | 75 | 415 | 330-6626 | yes | no | 0 | 166.7 | 28.34 | 148.3 | 122 | 12.61 |
| **5** | AL | 118 | 510 | 391-8027 | yes | no | 0 | 223.4 | 37.98 | 220.6 | 101 | 18.75 |
| **6** | MA | 121 | 510 | 355-9993 | no | yes | 24 | 218.2 | 37.09 | 348.5 | 108 | 29.62 |
| **7** | MO | 147 | 415 | 329-9001 | yes | no | 0 | 157.0 | 26.69 | 103.1 | 94 | 8.76 |
| **8** | LA | 117 | 408 | 335-4719 | no | no | 0 | 184.5 | 31.37 | 351.6 | 80 | 29.89 |
| **9** | WV | 141 | 415 | 330-8173 | yes | yes | 37 | 258.6 | 43.96 | 222.0 | 111 | 18.87 |
| **10** | IN | 65 | 415 | 329-6603 | no | no | 0 | 129.1 | 21.95 | 228.5 | 83 | 19.42 |
| **11** | RI | 74 | 415 | 344-9403 | no | no | 0 | 187.7 | 31.91 | 163.4 | 148 | 13.89 |
| **12** | IA | 168 | 408 | 363-1107 | no | no | 0 | 128.8 | 21.90 | 104.9 | 71 | 8.92 |
| **13** | MT | 95 | 510 | 394-8006 | no | no | 0 | 156.6 | 26.62 | 247.6 | 75 | 21.05 |
| **14** | IA | 62 | 415 | 366-9238 | no | no | 0 | 120.7 | 20.52 | 307.2 | 76 | 26.11 |
| **15** | NY | 161 | 415 | 351-7269 | no | no | 0 | 332.9 | 56.59 | 317.8 | 97 | 27.01 |
| **16** | ID | 85 | 408 | 350-8884 | no | yes | 27 | 196.4 | 33.39 | 280.9 | 90 | 23.88 |
| **17** | VT | 93 | 510 | 386-2923 | no | no | 0 | 190.7 | 32.42 | 218.2 | 111 | 18.55 |
| **18** | VA | 76 | 510 | 356-2992 | no | yes | 33 | 189.7 | 32.25 | 212.8 | 65 | 18.09 |
| **19** | TX | 73 | 415 | 373-2782 | no | no | 0 | 224.4 | 38.15 | 159.5 | 88 | 13.56 |
| **20** | FL | 147 | 415 | 396-5800 | no | no | 0 | 155.1 | 26.37 | 239.7 | 93 | 20.37 |
| **21** | CO | 77 | 408 | 393-7984 | no | no | 0 | 62.4 | 10.61 | 169.9 | 121 | 14.44 |
| **22** | AZ | 130 | 415 | 358-1958 | no | no | 0 | 183.0 | 31.11 | 72.9 | 99 | 6.20 |

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Charge | Eve Mins | Eve Calls | Eve Charge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **23** | SC | 111 | 415 | 350-2565 | no | no | 0 | 110.4 | 18.77 | 137.3 | 102 | 11.67 |
| **24** | VA | 132 | 510 | 343-4696 | no | no | 0 | 81.1 | 13.79 | 245.2 | 72 | 20.84 |
| **25** | NE | 174 | 415 | 331-3698 | no | no | 0 | 124.3 | 21.13 | 277.1 | 112 | 23.55 |
| **26** | WY | 57 | 408 | 357-3817 | no | yes | 39 | 213.0 | 36.21 | 191.1 | 112 | 16.24 |
| **27** | MT | 54 | 408 | 418-6412 | no | no | 0 | 134.3 | 22.83 | 155.5 | 100 | 13.22 |
| **28** | MO | 20 | 415 | 353-2630 | no | no | 0 | 190.0 | 32.30 | 258.2 | 84 | 21.95 |
| **29** | HI | 49 | 510 | 410-7789 | no | no | 0 | 119.3 | 20.28 | 215.1 | 109 | 18.28 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3303** | WI | 114 | 415 | 373-7308 | no | yes | 26 | 137.1 | 23.31 | 155.7 | 125 | 13.23 |
| **3304** | IL | 71 | 510 | 330-7137 | yes | no | 0 | 186.1 | 31.64 | 198.6 | 140 | 16.88 |
| **3305** | IN | 58 | 415 | 406-8445 | no | yes | 22 | 224.1 | 38.10 | 238.8 | 85 | 20.30 |
| **3306** | AL | 106 | 408 | 404-5283 | no | yes | 29 | 83.6 | 14.21 | 203.9 | 131 | 17.33 |
| **3307** | OK | 172 | 408 | 398-3632 | no | no | 0 | 203.9 | 34.66 | 234.0 | 123 | 19.89 |
| **3308** | IA | 45 | 415 | 399-5763 | no | no | 0 | 211.3 | 35.92 | 165.7 | 97 | 14.08 |
| **3309** | VT | 100 | 408 | 340-9449 | yes | no | 0 | 219.4 | 37.30 | 225.7 | 102 | 19.18 |
| **3310** | NY | 94 | 415 | 363-1123 | no | no | 0 | 190.4 | 32.37 | 92.0 | 107 | 7.82 |
| **3311** | LA | 128 | 415 | 361-2170 | no | no | 0 | 147.7 | 25.11 | 283.3 | 83 | 24.08 |
| **3312** | SC | 181 | 408 | 406-6304 | no | no | 0 | 229.9 | 39.08 | 144.4 | 93 | 12.27 |
| **3313** | ID | 127 | 408 | 392-5090 | no | no | 0 | 102.8 | 17.48 | 143.7 | 95 | 12.21 |
| **3314** | MO | 89 | 415 | 373-7713 | no | no | 0 | 178.7 | 30.38 | 233.7 | 74 | 19.86 |
| **3315** | ME | 149 | 415 | 392-1376 | no | yes | 18 | 148.5 | 25.25 | 114.5 | 106 | 9.73 |
| **3316** | MS | 103 | 510 | 390-6388 | no | yes | 29 | 164.1 | 27.90 | 219.1 | 96 | 18.62 |
| **3317** | SD | 163 | 415 | 379-7290 | yes | no | 0 | 197.2 | 33.52 | 188.5 | 113 | 16.02 |

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Charge | Eve Mins | Eve Calls | Eve Charge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3318** | OK | 52 | 415 | 397-9928 | no | no | 0 | 124.9 | 21.23 | 300.5 | 118 | 25.54 |
| **3319** | WY | 89 | 415 | 378-6924 | no | no | 0 | 115.4 | 19.62 | 209.9 | 115 | 17.84 |
| **3320** | GA | 122 | 510 | 411-5677 | yes | no | 0 | 140.0 | 23.80 | 196.4 | 77 | 16.69 |
| **3321** | VT | 60 | 415 | 400-2738 | no | no | 0 | 193.9 | 32.96 | 85.0 | 110 | 7.23 |
| **3322** | MD | 62 | 408 | 409-1856 | no | no | 0 | 321.1 | 54.59 | 265.5 | 122 | 22.57 |
| **3323** | IN | 117 | 415 | 362-5899 | no | no | 0 | 118.4 | 20.13 | 249.3 | 97 | 21.19 |
| **3324** | WV | 159 | 415 | 377-1164 | no | no | 0 | 169.8 | 28.87 | 197.7 | 105 | 16.80 |
| **3325** | OH | 78 | 408 | 368-8555 | no | no | 0 | 193.4 | 32.88 | 116.9 | 88 | 9.94 |
| **3326** | OH | 96 | 415 | 347-6812 | no | no | 0 | 106.6 | 18.12 | 284.8 | 87 | 24.21 |
| **3327** | SC | 79 | 415 | 348-3830 | no | no | 0 | 134.7 | 22.90 | 189.7 | 68 | 16.12 |
| **3328** | AZ | 192 | 415 | 414-4276 | no | yes | 36 | 156.2 | 26.55 | 215.5 | 126 | 18.32 |
| **3329** | WV | 68 | 415 | 370-3271 | no | no | 0 | 231.1 | 39.29 | 153.4 | 55 | 13.04 |
| **3330** | RI | 28 | 510 | 328-8230 | no | no | 0 | 180.8 | 30.74 | 288.8 | 58 | 24.55 |
| **3331** | CT | 184 | 510 | 364-6381 | yes | no | 0 | 213.8 | 36.35 | 159.6 | 84 | 13.57 |
| **3332** | TN | 74 | 415 | 400-4344 | no | yes | 25 | 234.4 | 39.85 | 265.9 | 82 | 22.60 |

3333 rows × 20 columns

In [7]:
```
#De forma alternativa, se puede hacer la diferencias por conjuntos

a = set(columnas_no_deseadas)
b = set(all_columnas)
sublist2 = list(b - a)
subset3 = data[sublist2]
subset3
```

Out[7]:

| | VMail Message | Account Length | Area Code | Phone | Intl Charge | Eve Charge | Night Charge | Day Charge | VMail Plan | Night Calls | Intl Mins |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 25 | 128 | 415 | 382-4657 | 2.70 | 16.78 | 11.01 | 45.07 | yes | 91 | 10.0 |
| **1** | 26 | 107 | 415 | 371-7191 | 3.70 | 16.62 | 11.45 | 27.47 | yes | 103 | 13.7 |
| **2** | 0 | 137 | 415 | 358-1921 | 3.29 | 10.30 | 7.32 | 41.38 | no | 104 | 12.2 |
| **3** | 0 | 84 | 408 | 375-9999 | 1.78 | 5.26 | 8.86 | 50.90 | no | 89 | 6.6 |
| **4** | 0 | 75 | 415 | 330-6626 | 2.73 | 12.61 | 8.41 | 28.34 | no | 121 | 10.1 |
| **5** | 0 | 118 | 510 | 391-8027 | 1.70 | 18.75 | 9.18 | 37.98 | no | 118 | 6.3 |
| **6** | 24 | 121 | 510 | 355-9993 | 2.03 | 29.62 | 9.57 | 37.09 | yes | 118 | 7.5 |
| **7** | 0 | 147 | 415 | 329-9001 | 1.92 | 8.76 | 9.53 | 26.69 | no | 96 | 7.1 |
| **8** | 0 | 117 | 408 | 335-4719 | 2.35 | 29.89 | 9.71 | 31.37 | no | 90 | 8.7 |
| **9** | 37 | 141 | 415 | 330-8173 | 3.02 | 18.87 | 14.69 | 43.96 | yes | 97 | 11.2 |
| **10** | 0 | 65 | 415 | 329-6603 | 3.43 | 19.42 | 9.40 | 21.95 | no | 111 | 12.7 |
| **11** | 0 | 74 | 415 | 344-9403 | 2.46 | 13.89 | 8.82 | 31.91 | no | 94 | 9.1 |
| **12** | 0 | 168 | 408 | 363-1107 | 3.02 | 8.92 | 6.35 | 21.90 | no | 128 | 11.2 |
| **13** | 0 | 95 | 510 | 394-8006 | 3.32 | 21.05 | 8.65 | 26.62 | no | 115 | 12.3 |
| **14** | 0 | 62 | 415 | 366-9238 | 3.54 | 26.11 | 9.14 | 20.52 | no | 99 | 13.1 |
| **15** | 0 | 161 | 415 | 351-7269 | 1.46 | 27.01 | 7.23 | 56.59 | no | 128 | 5.4 |
| **16** | 27 | 85 | 408 | 350-8884 | 3.73 | 23.88 | 4.02 | 33.39 | yes | 75 | 13.8 |
| **17** | 0 | 93 | 510 | 386-2923 | 2.19 | 18.55 | 5.83 | 32.42 | no | 121 | 8.1 |
| **18** | 33 | 76 | 510 | 356-2992 | 2.70 | 18.09 | 7.46 | 32.25 | yes | 108 | 10.0 |
| **19** | 0 | 73 | 415 | 373-2782 | 3.51 | 13.56 | 8.68 | 38.15 | no | 74 | 13.0 |
| **20** | 0 | 147 | 415 | 396-5800 | 2.86 | 20.37 | 9.40 | 26.37 | no | 133 | 10.6 |
| **21** | 0 | 77 | 408 | 393-7984 | 1.54 | 14.44 | 9.43 | 10.61 | no | 64 | 5.7 |
| **22** | 0 | 130 | 415 | 358-1958 | 2.57 | 6.20 | 8.18 | 31.11 | no | 78 | 9.5 |

| | VMail Message | Account Length | Area Code | Phone | Intl Charge | Eve Charge | Night Charge | Day Charge | VMail Plan | Night Calls | Intl Mins |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **23** | 0 | 111 | 415 | 350-2565 | 2.08 | 11.67 | 8.53 | 18.77 | no | 105 | 7.7 |
| **24** | 0 | 132 | 510 | 343-4696 | 2.78 | 20.84 | 10.67 | 13.79 | no | 115 | 10.3 |
| **25** | 0 | 174 | 415 | 331-3698 | 4.19 | 23.55 | 11.28 | 21.13 | no | 115 | 15.5 |
| **26** | 39 | 57 | 408 | 357-3817 | 2.57 | 16.24 | 8.22 | 36.21 | yes | 115 | 9.5 |
| **27** | 0 | 54 | 408 | 418-6412 | 3.97 | 13.22 | 4.59 | 22.83 | no | 68 | 14.7 |
| **28** | 0 | 20 | 415 | 353-2630 | 1.70 | 21.95 | 8.17 | 32.30 | no | 102 | 6.3 |
| **29** | 0 | 49 | 510 | 410-7789 | 3.00 | 18.28 | 8.04 | 20.28 | no | 90 | 11.1 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **3303** | 26 | 114 | 415 | 373-7308 | 3.11 | 13.23 | 11.14 | 23.31 | yes | 94 | 11.5 |
| **3304** | 0 | 71 | 510 | 330-7137 | 3.73 | 16.88 | 9.29 | 31.64 | no | 80 | 13.8 |
| **3305** | 22 | 58 | 415 | 406-8445 | 3.11 | 20.30 | 7.84 | 38.10 | yes | 86 | 11.5 |
| **3306** | 29 | 106 | 408 | 404-5283 | 2.19 | 17.33 | 10.33 | 14.21 | yes | 73 | 8.1 |
| **3307** | 0 | 172 | 408 | 398-3632 | 4.81 | 19.89 | 7.23 | 34.66 | no | 65 | 17.8 |
| **3308** | 0 | 45 | 415 | 399-5763 | 3.59 | 14.08 | 11.97 | 35.92 | no | 72 | 13.3 |
| **3309** | 0 | 100 | 408 | 340-9449 | 3.24 | 19.18 | 11.49 | 37.30 | no | 95 | 12.0 |
| **3310** | 0 | 94 | 415 | 363-1123 | 3.67 | 7.82 | 10.12 | 32.37 | no | 108 | 13.6 |
| **3311** | 0 | 128 | 415 | 361-2170 | 1.86 | 24.08 | 8.47 | 25.11 | no | 124 | 6.9 |
| **3312** | 0 | 181 | 408 | 406-6304 | 3.83 | 12.27 | 11.81 | 39.08 | no | 110 | 14.2 |
| **3313** | 0 | 127 | 408 | 392-5090 | 2.70 | 12.21 | 8.61 | 17.48 | no | 97 | 10.0 |
| **3314** | 0 | 89 | 415 | 373-7713 | 2.46 | 19.86 | 5.94 | 30.38 | no | 120 | 9.1 |
| **3315** | 18 | 149 | 415 | 392-1376 | 1.76 | 9.73 | 8.02 | 25.25 | yes | 98 | 6.5 |
| **3316** | 29 | 103 | 510 | 390-6388 | 3.32 | 18.62 | 9.91 | 27.90 | yes | 108 | 12.3 |
| **3317** | 0 | 163 | 415 | 379-7290 | 2.11 | 16.02 | 9.50 | 33.52 | no | 94 | 7.8 |

| | VMail Message | Account Length | Area Code | Phone | Intl Charge | Eve Charge | Night Charge | Day Charge | VMail Plan | Night Calls | Intl Mins |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **3318** | 0 | 52 | 415 | 397-9928 | 3.13 | 25.54 | 8.66 | 21.23 | no | 106 | 11.6 |
| **3319** | 0 | 89 | 415 | 378-6924 | 4.29 | 17.84 | 12.64 | 19.62 | no | 112 | 15.9 |
| **3320** | 0 | 122 | 510 | 411-5677 | 2.62 | 16.69 | 5.40 | 23.80 | no | 133 | 9.7 |
| **3321** | 0 | 60 | 415 | 400-2738 | 3.56 | 7.23 | 9.45 | 32.96 | no | 134 | 13.2 |
| **3322** | 0 | 62 | 408 | 409-1856 | 3.11 | 22.57 | 8.12 | 54.59 | no | 72 | 11.5 |
| **3323** | 0 | 117 | 415 | 362-5899 | 3.67 | 21.19 | 10.22 | 20.13 | no | 56 | 13.6 |
| **3324** | 0 | 159 | 415 | 377-1164 | 3.13 | 16.80 | 8.72 | 28.87 | no | 82 | 11.6 |
| **3325** | 0 | 78 | 408 | 368-8555 | 2.51 | 9.94 | 10.95 | 32.88 | no | 109 | 9.3 |
| **3326** | 0 | 96 | 415 | 347-6812 | 4.02 | 24.21 | 8.05 | 18.12 | no | 92 | 14.9 |
| **3327** | 0 | 79 | 415 | 348-3830 | 3.19 | 16.12 | 9.96 | 22.90 | no | 128 | 11.8 |
| **3328** | 36 | 192 | 415 | 414-4276 | 2.67 | 18.32 | 12.56 | 26.55 | yes | 83 | 9.9 |
| **3329** | 0 | 68 | 415 | 370-3271 | 2.59 | 13.04 | 8.61 | 39.29 | no | 123 | 9.6 |
| **3330** | 0 | 28 | 510 | 328-8230 | 3.81 | 24.55 | 8.64 | 30.74 | no | 91 | 14.1 |
| **3331** | 0 | 184 | 510 | 364-6381 | 1.35 | 13.57 | 6.26 | 36.35 | no | 137 | 5.0 |
| **3332** | 25 | 74 | 415 | 400-4344 | 3.70 | 22.60 | 10.86 | 39.85 | yes | 77 | 13.7 |

3333 rows × 20 columns

## Seleccionar "n" filas para un nuevo subset

In [8]:
```
# Se considera del límite inferior hasta 1 menos del límite superior del rango en el corchete
data[10:14]
```

Out[8]:

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Calls | Day Charge | ... | Eve Calls | Ev Charg |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **10** | IN | 65 | 415 | 329-6603 | no | no | 0 | 129.1 | 137 | 21.95 | ... | 83 | 19.4 |
| **11** | RI | 74 | 415 | 344-9403 | no | no | 0 | 187.7 | 127 | 31.91 | ... | 148 | 13.8 |
| **12** | IA | 168 | 408 | 363-1107 | no | no | 0 | 128.8 | 96 | 21.90 | ... | 71 | 8.9 |
| **13** | MT | 95 | 510 | 394-8006 | no | no | 0 | 156.6 | 88 | 26.62 | ... | 75 | 21.0 |

4 rows × 21 columns

In [9]:
```
# Desde una fila en adelante

data[3330:]
```

Out[9]:

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Calls | Day Charge | ... | Eve Calls | Cha |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **3330** | RI | 28 | 510 | 328-8230 | no | no | 0 | 180.8 | 109 | 30.74 | ... | 58 | 24 |
| **3331** | CT | 184 | 510 | 364-6381 | yes | no | 0 | 213.8 | 105 | 36.35 | ... | 84 | 1 |
| **3332** | TN | 74 | 415 | 400-4344 | no | yes | 25 | 234.4 | 113 | 39.85 | ... | 82 | 2 |

3 rows × 21 columns

In [10]:
```
# Hasta una fila hacia atráz

data[:8]
```

Out[10]:

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Calls | Day Charge | ... | Eve Calls | Eve Charge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KS | 128 | 415 | 382-4657 | no | yes | 25 | 265.1 | 110 | 45.07 | ... | 99 | 16.78 |
| **1** | OH | 107 | 415 | 371-7191 | no | yes | 26 | 161.6 | 123 | 27.47 | ... | 103 | 16.62 |
| **2** | NJ | 137 | 415 | 358-1921 | no | no | 0 | 243.4 | 114 | 41.38 | ... | 110 | 10.30 |
| **3** | OH | 84 | 408 | 375-9999 | yes | no | 0 | 299.4 | 71 | 50.90 | ... | 88 | 5.26 |
| **4** | OK | 75 | 415 | 330-6626 | yes | no | 0 | 166.7 | 113 | 28.34 | ... | 122 | 12.61 |
| **5** | AL | 118 | 510 | 391-8027 | yes | no | 0 | 223.4 | 98 | 37.98 | ... | 101 | 18.75 |
| **6** | MA | 121 | 510 | 355-9993 | no | yes | 24 | 218.2 | 88 | 37.09 | ... | 108 | 29.62 |
| **7** | MO | 147 | 415 | 329-9001 | yes | no | 0 | 157.0 | 79 | 26.69 | ... | 94 | 8.76 |

8 rows × 21 columns

In [11]:
```
# Usuarios con Day Mins > 300

data1 = data[data['Day Mins'] > 300]
data1.shape
```

Out[11]: (43, 21)

In [12]:
```
# Usuarios de Nueva York (State = 'NY')
data2 = data[data['State'] == 'NY']
data2.shape
```

Out[12]: (83, 21)

In [13]:
```
# Más de una condición a cumplir
# AND -> &
# Buscamos los usuarios que cumplan con 2 condiciones, que hablen más de 300 m
inutos y sean de New York
data3 = data[(data['Day Mins'] > 300) & (data['State'] == 'NY')]
data3.shape
```

Out[13]: (2, 21)

Interesante, sólo dos usuarios!!!

¿Qué ocurre si cambiamos la condición por un "OR"? Pues bien, deberíamos tener un conjunto más grande de datos

In [14]:
```
# Más de una condición a cumplir
# OR -> |
# Buscamos los usuarios que cumplan con; que hablen más de 300 minutos ó bien
 que sean de New York

data4 = data[(data['Day Mins'] > 300) | (data['State'] == 'NY')]
data4.shape
```

Out[14]: (124, 21)

¿Se puede obtener un subset donde se combinen condiciones sobre filas y columnas? La respuesta es sí, vamos a por ello!

# Subconjuntos con loc e iloc y creación de nuevas columnas

Se pide obtener los valores de los 50 primeros individuos con respecto a minutos del día, de noche y longitud de la cuenta

In [15]:
```
# Primera forma

subset_first_50 = data[['Day Mins', 'Night Mins', 'Account Length']][:50]
subset_first_50.head()
```

Out[15]:

| | Day Mins | Night Mins | Account Length |
|---|---|---|---|
| **0** | 265.1 | 244.7 | 128 |
| **1** | 161.6 | 254.4 | 107 |
| **2** | 243.4 | 162.6 | 137 |
| **3** | 299.4 | 196.9 | 84 |
| **4** | 166.7 | 186.9 | 75 |

In [16]:
```
# Segunda forma, con iloc

data.iloc[1:10, 3:6] # Las filas entre la 1 a la 10, y las columnas en las pos
iciones 3 a la 6 (la 6 sin incluir)
```

Out[16]:

| | Phone | Int'l Plan | VMail Plan |
|---|---|---|---|
| **1** | 371-7191 | no | yes |
| **2** | 358-1921 | no | no |
| **3** | 375-9999 | yes | no |
| **4** | 330-6626 | yes | no |
| **5** | 391-8027 | yes | no |
| **6** | 355-9993 | no | yes |
| **7** | 329-9001 | yes | no |
| **8** | 335-4719 | no | no |
| **9** | 330-8173 | yes | yes |

In [17]:
```
#Más ejemplos
data.iloc[:, 3:6] # Todas las filas para las columnas de la 3 a la 6
data.iloc[1:10, :] # Todas las columnas, para los registros del 1 al 10
```

Out[17]:

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Calls | Day Charge | ... | Eve Calls | Eve Charge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | OH | 107 | 415 | 371-7191 | no | yes | 26 | 161.6 | 123 | 27.47 | ... | 103 | 16.62 |
| **2** | NJ | 137 | 415 | 358-1921 | no | no | 0 | 243.4 | 114 | 41.38 | ... | 110 | 10.30 |
| **3** | OH | 84 | 408 | 375-9999 | yes | no | 0 | 299.4 | 71 | 50.90 | ... | 88 | 5.26 |
| **4** | OK | 75 | 415 | 330-6626 | yes | no | 0 | 166.7 | 113 | 28.34 | ... | 122 | 12.61 |
| **5** | AL | 118 | 510 | 391-8027 | yes | no | 0 | 223.4 | 98 | 37.98 | ... | 101 | 18.75 |
| **6** | MA | 121 | 510 | 355-9993 | no | yes | 24 | 218.2 | 88 | 37.09 | ... | 108 | 29.62 |
| **7** | MO | 147 | 415 | 329-9001 | yes | no | 0 | 157.0 | 79 | 26.69 | ... | 94 | 8.76 |
| **8** | LA | 117 | 408 | 335-4719 | no | no | 0 | 184.5 | 97 | 31.37 | ... | 80 | 29.89 |
| **9** | WV | 141 | 415 | 330-8173 | yes | yes | 37 | 258.6 | 84 | 43.96 | ... | 111 | 18.87 |

9 rows × 21 columns

In [18]:
```
# Usando loc, para ingresar los nombres de las columnas

data.loc[:10, ['Day Mins', 'Night Mins', 'Account Length']]
```

Out[18]:

| | Day Mins | Night Mins | Account Length |
|---|---|---|---|
| **0** | 265.1 | 244.7 | 128 |
| **1** | 161.6 | 254.4 | 107 |
| **2** | 243.4 | 162.6 | 137 |
| **3** | 299.4 | 196.9 | 84 |
| **4** | 166.7 | 186.9 | 75 |
| **5** | 223.4 | 203.9 | 118 |
| **6** | 218.2 | 212.6 | 121 |
| **7** | 157.0 | 211.8 | 147 |
| **8** | 184.5 | 215.8 | 117 |
| **9** | 258.6 | 326.4 | 141 |
| **10** | 129.1 | 208.8 | 65 |

Crear una nueva columna

In [19]:
```
# Aquí un ejemplo para agregar una columna con los minutos totales

data['Total Mins'] = data['Day Mins'] + data['Night Mins'] + data['Eve Mins']
data['Total Mins'].head()
```

Out[19]:
```
0    707.2
1    611.5
2    527.2
3    558.2
4    501.9
Name: Total Mins, dtype: float64
```

In [20]:
```
# Otro ejemplo para agregar una columna con las llamadas totales
data['Total Calls'] = data['Day Calls'] + data['Night Calls'] + data['Eve Call
s']
data['Total Calls'].head()
```

Out[20]:
```
0    300
1    329
2    328
3    248
4    356
Name: Total Calls, dtype: int64
```

In [21]:
```
data.shape
```

Out[21]:
```
(3333, 23)
```

In [22]:
```
data.head()
```

Out[22]:

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Calls | Day Charge | ... | Night Mins | Night Calls |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KS | 128 | 415 | 382-4657 | no | yes | 25 | 265.1 | 110 | 45.07 | ... | 244.7 | 91 |
| **1** | OH | 107 | 415 | 371-7191 | no | yes | 26 | 161.6 | 123 | 27.47 | ... | 254.4 | 103 |
| **2** | NJ | 137 | 415 | 358-1921 | no | no | 0 | 243.4 | 114 | 41.38 | ... | 162.6 | 104 |
| **3** | OH | 84 | 408 | 375-9999 | yes | no | 0 | 299.4 | 71 | 50.90 | ... | 196.9 | 89 |
| **4** | OK | 75 | 415 | 330-6626 | yes | no | 0 | 166.7 | 113 | 28.34 | ... | 186.9 | 121 |

5 rows × 23 columns

Podemos evidenciar que creamos las dos columnas al final de nuestro dataframe

# Generación aleatoria de números

In [23]:
```
#Generar 10 números aleatorios con random
i=0
while i < 10:
    print(np.random.randint(1,100))
    i+=1
```

```
1
21
34
47
37
91
98
2
31
71
```

In [24]:
```
## La forma más clásica de generar un número aleatorio es entre 0 y 1.
## Se generan 10 números
i=0
while i < 10:
    print(np.random.random())
    i+=1
```

```
0.3792106797823498
0.39990153763983916
0.08558157363440766
0.8286610276578852
0.5429550987016851
0.47842378849684064
0.2422158395336389
0.0879273909248035
0.5279648605320951
0.4162943419136004
```

In [25]:
```
## Existe una librería que genera una lista de número de forma aleatoria dentr
o de un rango

import random

# Imprimir 10 números

for i in range(10):
    print(random.randrange(1, 100))
```

```
18
13
17
24
94
13
45
56
55
31
```

**Shuffling -> "es como barajar un mazo de cartas, altera el orden"**

In [26]:
```
a = np.arange(100)
a
```

Out[26]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84,
       85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99])
```

In [27]:
```
np.random.shuffle(a)
a
```

Out[27]:
```
array([12, 87, 44, 92,  3,  9, 13, 96, 93, 61, 67, 95, 34,  1, 19, 68, 94,
       26,  2,  6, 72,  5, 42, 82, 27, 30, 89, 90, 74, 99, 98, 51, 20, 88,
       43, 58, 60, 65, 62, 77, 29, 64,  7, 52, 81, 31, 36, 54, 85, 22, 33,
       55, 32, 10, 18, 71, 79, 40, 97, 53,  0, 80,  8, 23, 76, 17, 70, 83,
       14, 16, 25, 24, 86, 56, 37, 45, 63, 59, 41, 38, 91, 73, 78, 21, 50,
       69, 28, 84, 35, 46, 11, 48, 75, 15, 57, 39, 47, 49,  4, 66])
```

### La semilla -> Seed

In [28]:
```
## Fijamos una semilla

np.random.seed(2020)
for i in range(5):
    print(np.random.random())
```

```
0.9862768288615988
0.8733919458206546
0.5097455249715815
0.27183571428207576
0.33691872774596354
```

¿Para qué sirve? Para replicar los resultados obtenidos, es muy importante que se establezca al principio del trabajo.

Antes de comenzar un experimento, se recomienda establecer una semilla

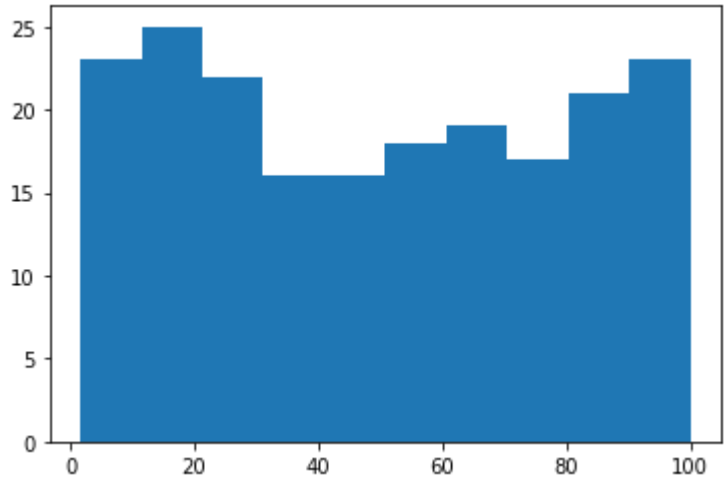
## Funciones de distribución de probabilidades

### Distribución uniforme

In [29]:
```
## Librerías

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

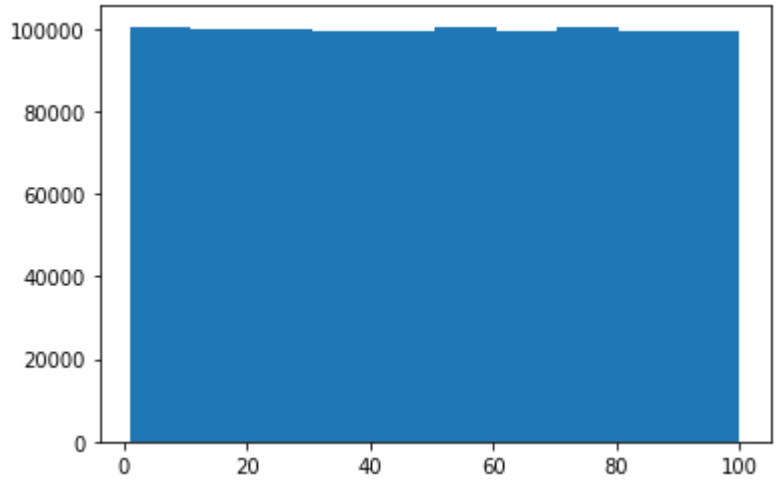
In [30]:
```
# Generamos  n muestras entre a y b, distribuidas de forma uniforme

a = 1 # Límite inferior
b = 100 # Límite superior
n = 200 # Cantidad de muestras a generar
data = np.random.uniform(a,b,n)
```

In [31]:
```
# Graficamos los datos generados

plt.hist(data);
```



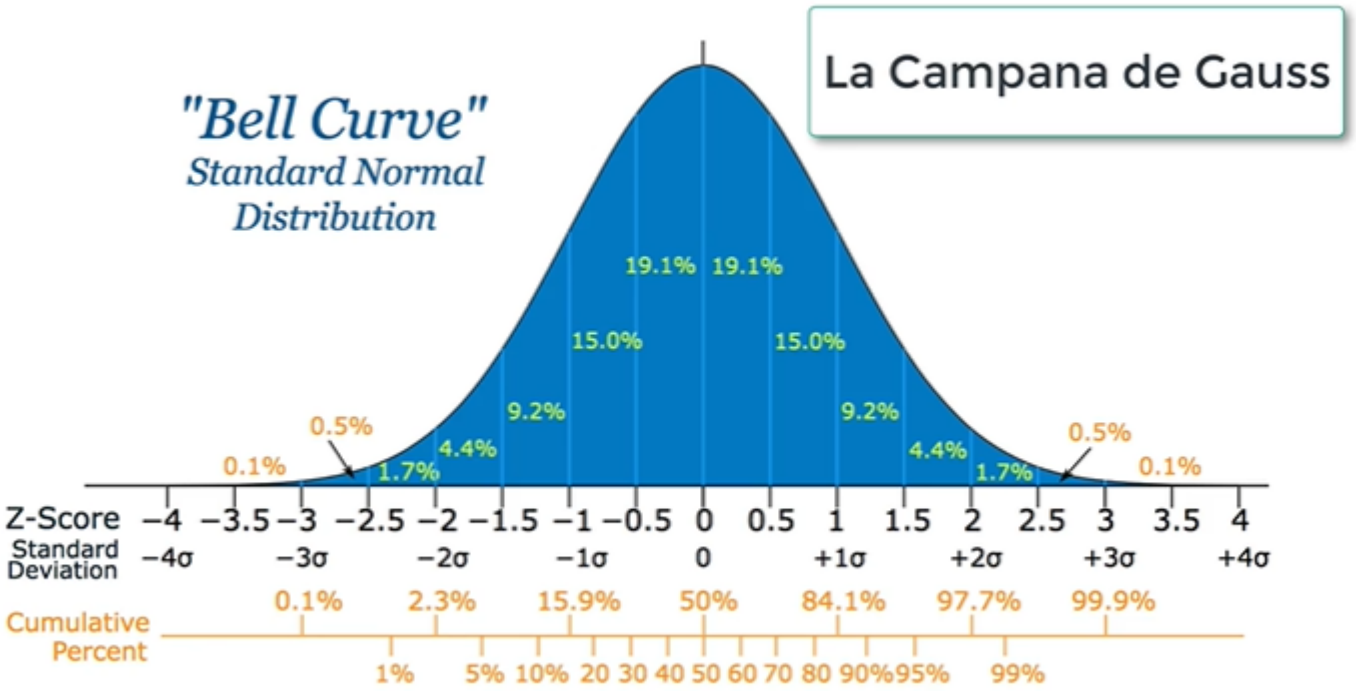

No parece tan uniforme, ¿por qué? Porque tomamos muy pocas muestras, probemos con 1.000.000 de muestras

In [32]:
```
n = 1000000 # Cantidad de muestras a generar
data2 = np.random.uniform(a,b,n)

# Graficamos los datos generados

plt.hist(data2);
```

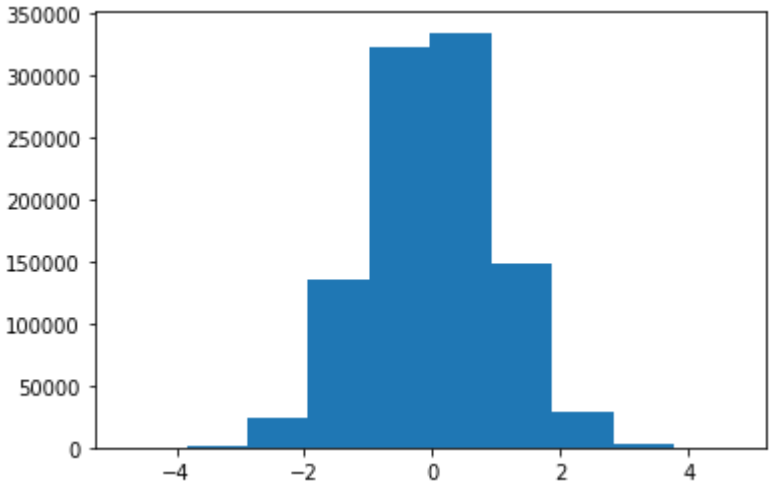



Ahora si parece una distribución uniforme !! :)

## Distribución Normal

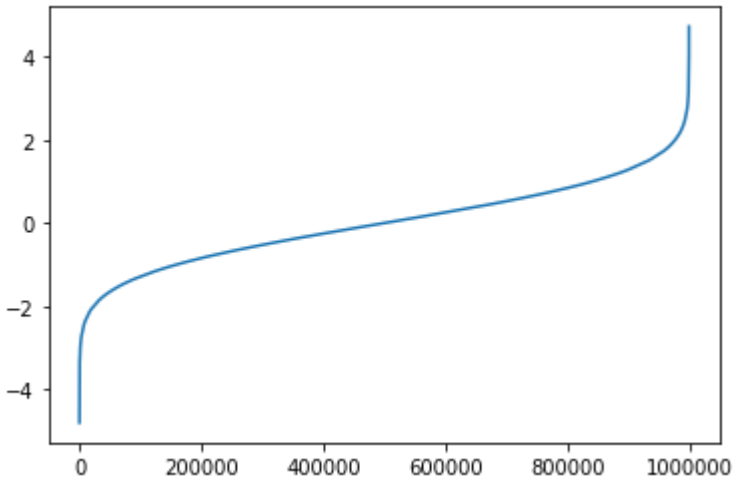



In [33]:
```
n = 1000000 # Cantidad de muestras a generar
data3 = np.random.randn(n) # Generador de muestras

# Graficamos los datos generados
plt.hist(data3);
```



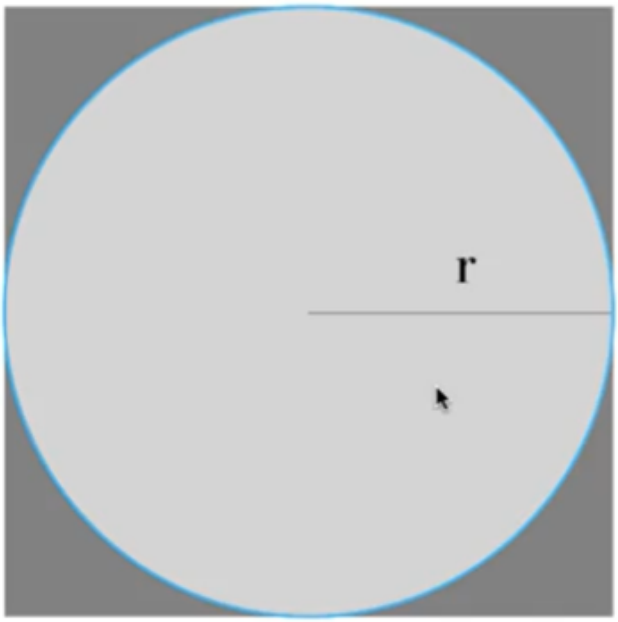
In [34]:
```
# Graficar la función de distribución acumulada

x = range(1, 1000001) # Generar el eje x
plt.plot(x, sorted(data3))
```

Out[34]:
```
[<matplotlib.lines.Line2D at 0x230bae89518>]
```



# La simulación de Monte Carlo

### ¿Cuál es la probabilidad de que al elijir un punto cualquiera al azar dentro del cuadrado, caiga dentro del círculo?




$$P(\text{caer en el círculo}) = \frac{\text{caer dentro del círculo}}{\text{caer dentro del cuadrado}} = \frac{\text{área círculo}}{\text{área cuadrado}} = \frac{\pi \times r \times r}{2 \times r \times 2 \times r} = \frac{\pi}{4}$$

### Y bien, si multiplicamos a esta probabilidad por $4$ se puede obtener una

Objetivo = simular el cálculo del valor de $\pi$

Procedimiento:

1. Generamos dos números aleatorios $x$ e $y$ entre 0 y 1 (tienen que ser equiprobables)
2. Calculamos $x^2 + y^2$:

    2.1. Si el valor es inferior a 1 $\rightarrow$ estamos dentro del círculo.

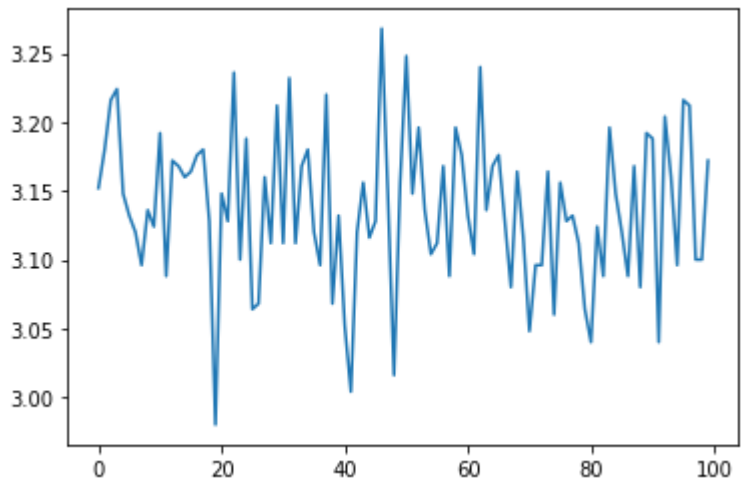
    2.2. Si el valor es superior a 1 $\rightarrow$ estamos fuera del círculo.

1. Calculamos el número total de veces que están dentro del círculo y lo dividimos entre el número total de intentos para obtener una aproximación de la probabilidad de caer dentro del círculo
2. Usamos dicha probabilidad para aproximar el valor de $\pi$.
3. Repetimos el experimento un número n de veces, para obtener diferentes aproximaciones de $\pi$.
4. Calculamos el promedio de los experimentos anteriores para dar un valor final de $\pi$.

In [35]:
```
# Generamos una variable que guarde los valores de pi obtenidos
pi_avg = 0
n = 1000 # cantidad de muestras a generar
pi_value_list =[] # guardar los valores de pi obtenidos
m = 100 # cantidad de experimentos  realizar

for i in range(m): # Calcular el experimento m veces
    value = 0 # para guardar el número de veces que acertamos dentro del círcu
lo
    x = np.random.uniform(0,1,n).tolist() # Valores uniformes guardados en una
lista, para x
    y = np.random.uniform(0,1,n).tolist() # Valores uniformes guardados en una
lista, para y
    for j in range(n): # Hacer un recorrido para obtener el valor x cuadrado m
ás y cuadrado
        z = x[j]*x[j] + y[j]*y[j]
        if z<=1:
            value+=1 # Incrementamos en uno el número de casos favorables
    float_value = float(value)
    pi_value = float_value * 4 / n # Valor de casos favorables dividido en cas
os totales
    pi_value_list.append(pi_value) # Agregar a la lista un nuevo valor obtenid
o de pi
    pi_avg += pi_value # Acumular el valor de pi

pi = pi_avg/m # Obtener el promedio de todos los experimentos
print(pi) # Mostrar el valor de pi obtenido
plt.plot(pi_value_list) # Graficar los valores de pi obtenidos
```

```
3.1376000000000017
```

Out[35]:
```
[<matplotlib.lines.Line2D at 0x230b8699358>]
```



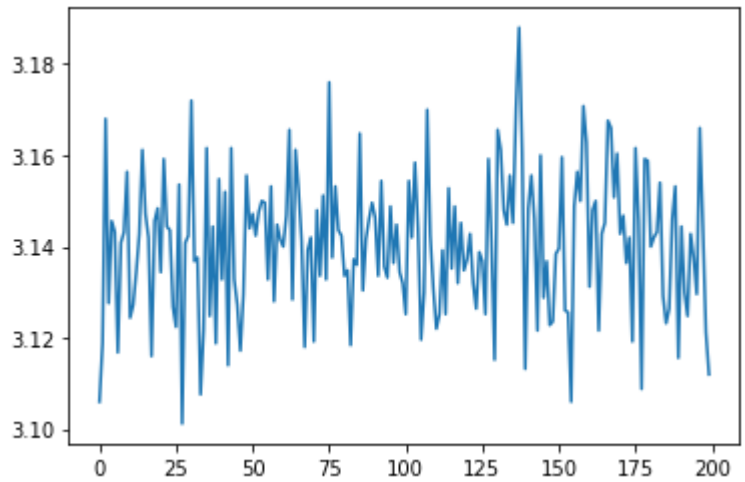

¿Y si creamos una función que realice este experimento? Let's do it!

In [36]:
```
def pi_montecarlo(n=1000, n_exp=100):
    # Generamos una variable que guarde los valores de pi obtenidos
    pi_avg = 0
    pi_value_list =[] # guardar los valores de pi obtenidos
    m = n_exp # cantidad de experimentos  realizar

    for i in range(m): # Calcular el experimento m veces
        value = 0 # para guardar el número de veces que acertamos dentro del c
írculo
        x = np.random.uniform(0,1,n).tolist() # n Valores uniformes guardados
 en una lista, para x
        y = np.random.uniform(0,1,n).tolist() # n Valores uniformes guardados
 en una lista, para y
        for j in range(n): # Hacer un recorrido para obtener el valor x cuadra
do más y cuadrado
            z = x[j]*x[j] + y[j]*y[j]
            if z<=1:
                value+=1 # Incrementamos en uno el número de casos favorables
        float_value = float(value)
        pi_value = float_value * 4 / n # Valor de casos favorables dividido en
casos totales
        pi_value_list.append(pi_value) # Agregar a la lista un nuevo valor obt
enido de pi
        pi_avg += pi_value # Acumular el valor de pi

    pi = pi_avg/m # Obtener el promedio de todos los experimentos
    print(pi) # Mostrar el valor de pi obtenido
    fig = plt.plot(pi_value_list) # Graficar los valores de pi obtenidos
    return (pi, fig)
```

In [37]:
```
pi_montecarlo(10000, 200)
```

```
3.140576
```

Out[37]:
```
(3.140576, [<matplotlib.lines.Line2D at 0x230b6e1b898>])
```




Sensacional!! Continuemos con la generación de data sets "dummy"

## Dummy Data sets

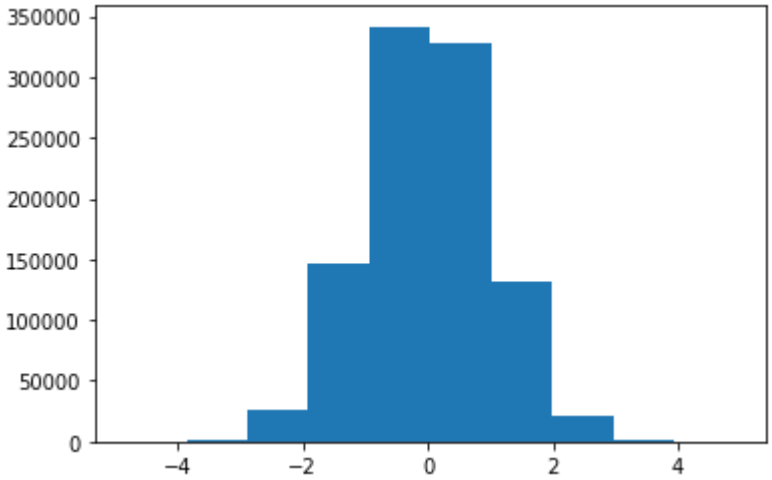
In [38]:
```
# Construyamos un dataframe con todo lo que hemos aprendido sobre distribucion
es

# Cantidad de muestras
n = 1000000

data_1 = pd.DataFrame(
    {
     'A' : np.random.randn(n), # Distribución normal (0,1)
     'B' : 1.5 + 2.5*np.random.randn(n), # Distribución normal (1.5, 2.5)
     'C' : np.random.uniform(5, 32, n) # Distribución Uniforme entre 5 y 32
    }
)
# Para todas las variables, se generó n muestras. Revisemos
data_1.describe()
```

Out[38]:

| | A | B | C |
|---|---|---|---|
| **count** | 1000000.000000 | 1000000.000000 | 1000000.000000 |
| **mean** | 0.000387 | 1.497605 | 18.512680 |
| **std** | 1.000015 | 2.499402 | 7.797526 |
| **min** | -4.827719 | -10.056537 | 5.000075 |
| **25%** | -0.673379 | -0.190753 | 11.762609 |
| **50%** | -0.000087 | 1.500650 | 18.504083 |
| **75%** | 0.674366 | 3.182166 | 25.271306 |
| **max** | 4.913072 | 13.713005 | 31.999993 |

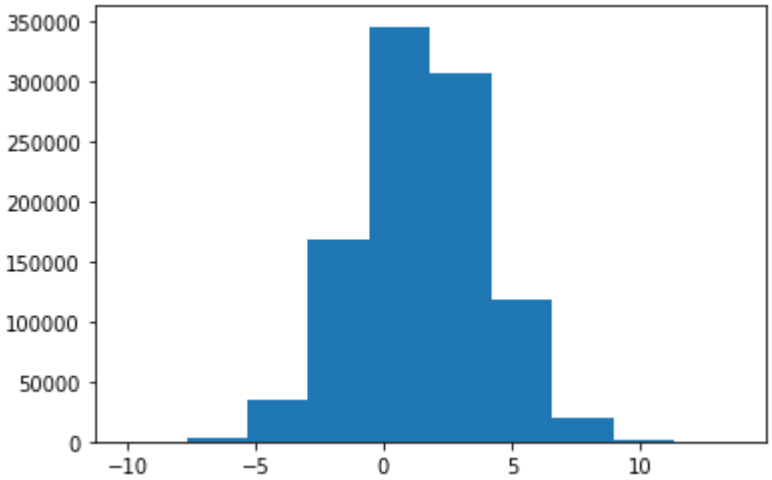
In [39]:
```
# Revisemos las distrbuciones

# Distribución normal (0,1)
plt.hist(data_1['A'])
```

Out[39]:
```
(array([5.00000e+01, 1.93300e+03, 2.63880e+04, 1.47231e+05, 3.41541e+05,
        3.28130e+05, 1.31417e+05, 2.17890e+04, 1.47800e+03, 4.30000e+01]),
 array([-4.82771871, -3.85363966, -2.8795606 , -1.90548154, -0.93140249,
         0.04267657,  1.01675562,  1.99083468,  2.96491374,  3.93899279,
         4.91307185]),
 <a list of 10 Patch objects>)
```



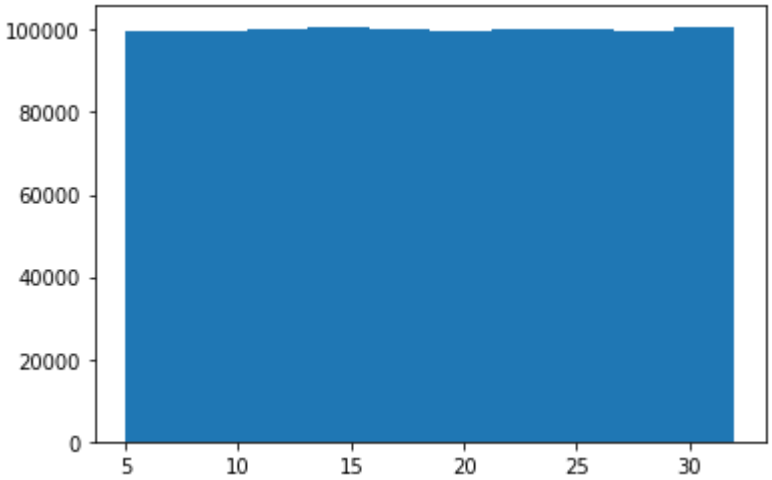

In [40]:
```
# Distribución normal (1.5, 2.5)
plt.hist(data_1['B'])
```

Out[40]:
```
(array([1.00000e+02, 3.10600e+03, 3.51070e+04, 1.68267e+05, 3.45984e+05,
        3.07824e+05, 1.18721e+05, 1.94850e+04, 1.36300e+03, 4.30000e+01]),
 array([-10.0565369 ,  -7.6795827 ,  -5.30262849,  -2.92567428,
         -0.54872008,   1.82823413,   4.20518834,   6.58214254,
          8.95909675,  11.33605096,  13.71300516]),
 <a list of 10 Patch objects>)
```



In [41]:
```
# Distribución Uniforme entre 5 y 32
plt.hist(data_1['C'])
```

Out[41]:
```
(array([ 99717.,  99855.,  99967., 100400.,  99924.,  99536., 100000.,
         99921.,  99904., 100776.]),
 array([ 5.00007513,  7.70006693, 10.40005873, 13.10005054, 15.80004234,
        18.50003414, 21.20002594, 23.90001775, 26.60000955, 29.30000135,
        31.99999315]),
 <a list of 10 Patch objects>)
```




In [42]:
```
data_2 = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/customer-churn-model/Customer Churn Model.txt')
data_2.head()
```

Out[42]:

| | State | Account Length | Area Code | Phone | Int'l Plan | VMail Plan | VMail Message | Day Mins | Day Calls | Day Charge | ... | Eve Calls | Eve Charge |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KS | 128 | 415 | 382-4657 | no | yes | 25 | 265.1 | 110 | 45.07 | ... | 99 | 16.78 |
| **1** | OH | 107 | 415 | 371-7191 | no | yes | 26 | 161.6 | 123 | 27.47 | ... | 103 | 16.62 |
| **2** | NJ | 137 | 415 | 358-1921 | no | no | 0 | 243.4 | 114 | 41.38 | ... | 110 | 10.30 |
| **3** | OH | 84 | 408 | 375-9999 | yes | no | 0 | 299.4 | 71 | 50.90 | ... | 88 | 5.26 |
| **4** | OK | 75 | 415 | 330-6626 | yes | no | 0 | 166.7 | 113 | 28.34 | ... | 122 | 12.61 |

5 rows × 21 columns

In [43]:
```
colum_names = data_2.columns.values.tolist()
a = len(colum_names)
a
```

Out[43]:
```
21
```

In [44]:
```
new_data = pd.DataFrame (
    {
        'Column Name': colum_names,
        'A' : np.random.randn(a),
        'B' : np.random.uniform(0,1,a)
    }
)

new_data
```

Out[44]:

| | Column Name | A | B |
|---|---|---|---|
| **0** | State | 0.018997 | 0.209223 |
| **1** | Account Length | 0.140282 | 0.711497 |
| **2** | Area Code | -0.280073 | 0.710835 |
| **3** | Phone | -0.888329 | 0.476497 |
| **4** | Int'l Plan | 2.091324 | 0.341038 |
| **5** | VMail Plan | 0.569381 | 0.472723 |
| **6** | VMail Message | -0.605941 | 0.256080 |
| **7** | Day Mins | 0.447846 | 0.129207 |
| **8** | Day Calls | 0.699854 | 0.448251 |
| **9** | Day Charge | 0.046414 | 0.837945 |
| **10** | Eve Mins | 1.681305 | 0.163688 |
| **11** | Eve Calls | -0.639666 | 0.798479 |
| **12** | Eve Charge | 1.098624 | 0.288328 |
| **13** | Night Mins | -1.138158 | 0.438249 |
| **14** | Night Calls | -1.883176 | 0.983024 |
| **15** | Night Charge | 0.366658 | 0.240144 |
| **16** | Intl Mins | 1.768024 | 0.241438 |
| **17** | Intl Calls | 1.998162 | 0.700124 |
| **18** | Intl Charge | 1.695546 | 0.881544 |
| **19** | CustServ Calls | -0.019813 | 0.159834 |
| **20** | Churn? | 0.002418 | 0.476904 |

# Dummy Data Sets con datos categóricos

In [45]:
```
# Generamos dos variables categóricas

gender = ['Mujer', 'Hombre']
income = ['Pobre', 'Clase media', 'Rico']

n = 500 # Cantidad de muestras

# Creamos las listas para guardar los datos aleatorios

gender_data = []
income_data = []

# Ciclo para obtener n muestras sobre nuestras variables
for i in range(0, 500):
    gender_data.append(np.random.choice(gender))
    income_data.append(np.random.choice(income))

# Revisemos los primeros 10 registros
gender_data[:10]
```

Out[45]:
```
['Hombre',
 'Hombre',
 'Hombre',
 'Hombre',
 'Mujer',
 'Mujer',
 'Mujer',
 'Mujer',
 'Mujer',
 'Mujer']
```

In [46]:
```
# Revisemos los primeros 10 registros
income_data[:10]
```

Out[46]:
```
['Rico',
 'Clase media',
 'Pobre',
 'Clase media',
 'Clase media',
 'Clase media',
 'Rico',
 'Rico',
 'Rico',
 'Clase media']
```

In [47]:
```
# Generamos nuevas variables, pero ahora numéricas
# Con distribuciones normales
# Z -> N(0,1)

# N(m, s) -> m + s *z
height = 160 + 30 * np.random.randn(n)
weight = 65 + 25 * np.random.randn(n)
age = 30 + 12 * np.random.randn(n)
income = 18000 + 3500 * np.random.randn(n)
```

In [48]:
```
# Creamos el dataframe

data_3 = pd.DataFrame(
    {
        'Sexo': gender_data,
        'Estado Economico': income_data,
        'Peso': weight,
        'Altura': height,
        'Edad': age,
        'Ingresos': income
    }
)

data_3.head()
```

Out[48]:

| | Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|---|
| **0** | Hombre | Rico | 33.346201 | 178.854874 | 45.385946 | 20813.470719 |
| **1** | Hombre | Clase media | 113.361591 | 141.806900 | 13.956049 | 15429.476751 |
| **2** | Hombre | Pobre | 61.085082 | 188.536748 | 23.820786 | 13117.313909 |
| **3** | Hombre | Clase media | 18.778331 | 140.283566 | 42.871327 | 17759.165610 |
| **4** | Mujer | Clase media | 61.181933 | 121.918602 | 13.601447 | 23901.455692 |

## Agrupación de datos

In [49]:
```
agrupados_sexo = data_3.groupby('Sexo')
agrupados_sexo.groups
```

Out[49]:
```
{'Hombre': Int64Index([  0,   1,   2,   3,  11,  12,  18,  20,  21,  23,
                ...
                484, 486, 487, 488, 491, 492, 494, 496, 497, 498],
               dtype='int64', length=257),
 'Mujer': Int64Index([  4,   5,   6,   7,   8,   9,  10,  13,  14,  15,
                ...
                477, 478, 479, 480, 485, 489, 490, 493, 495, 499],
               dtype='int64', length=243)}
```

In [50]:
```
# Revisemos los datos agrupados

for nombre, grupo in agrupados_sexo:
    print(nombre)
    print(grupo)
```

Hombre

| | Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|---|
| 0 | Hombre | Rico | 33.346201 | 178.854874 | 45.385946 | 20813.470719 |
| 1 | Hombre | Clase media | 113.361591 | 141.806900 | 13.956049 | 15429.476751 |
| 2 | Hombre | Pobre | 61.085082 | 188.536748 | 23.820786 | 13117.313909 |
| 3 | Hombre | Clase media | 18.778331 | 140.283566 | 42.871327 | 17759.165610 |
| 11 | Hombre | Clase media | 75.303962 | 154.932648 | 34.281199 | 19046.369405 |
| 12 | Hombre | Clase media | 62.728649 | 166.651245 | 29.479539 | 17420.128140 |
| 18 | Hombre | Clase media | 84.737532 | 182.552952 | 38.527954 | 20331.740509 |
| 20 | Hombre | Clase media | 71.459400 | 145.489352 | 25.602147 | 22406.519566 |
| 21 | Hombre | Rico | 48.845505 | 171.255861 | 36.341825 | 17866.847612 |
| 23 | Hombre | Clase media | 79.984957 | 171.061312 | 30.883470 | 21461.458889 |
| 24 | Hombre | Rico | 69.516723 | 161.285463 | 14.448105 | 16987.956652 |
| 25 | Hombre | Clase media | 62.429489 | 104.401969 | 41.774141 | 14537.604535 |
| 27 | Hombre | Clase media | 58.603480 | 208.998805 | 28.943294 | 24312.776440 |
| 31 | Hombre | Clase media | 72.877127 | 159.872874 | 36.165485 | 18896.238479 |
| 32 | Hombre | Rico | 60.457446 | 108.682354 | 32.892411 | 18915.110284 |
| 33 | Hombre | Clase media | 60.016595 | 129.018628 | 23.147194 | 20817.312735 |
| 34 | Hombre | Rico | 56.810920 | 131.873287 | 53.618540 | 14463.387698 |
| 36 | Hombre | Rico | 86.511228 | 173.451385 | 30.681507 | 24208.054556 |
| 37 | Hombre | Rico | 65.079106 | 130.523457 | 31.906173 | 16176.374604 |
| 40 | Hombre | Pobre | 77.669863 | 98.649087 | 12.787209 | 17043.917381 |
| 41 | Hombre | Pobre | 69.902331 | 185.231122 | 23.337743 | 14492.387956 |
| 42 | Hombre | Pobre | 56.935984 | 192.875052 | 10.721323 | 18044.052640 |
| 45 | Hombre | Rico | 98.355855 | 177.743133 | 42.185538 | 17108.076404 |
| 52 | Hombre | Rico | 16.357960 | 161.415063 | 23.518715 | 17956.664940 |
| 54 | Hombre | Rico | 90.269989 | 164.009535 | 40.801611 | 19810.412952 |
| 55 | Hombre | Pobre | 80.401239 | 125.134461 | 43.219089 | 19964.814527 |
| 58 | Hombre | Pobre | 72.370506 | 186.351808 | 31.013231 | 18443.760802 |
| 61 | Hombre | Rico | 119.871572 | 139.603747 | 17.506358 | 21872.829970 |
| 62 | Hombre | Pobre | 82.419144 | 174.803364 | 16.645850 | 11925.884175 |
| 65 | Hombre | Rico | 91.592110 | 207.311949 | 43.121437 | 22271.013418 |
| .. | ... | ... | ... | ... | ... | ... |
| 442 | Hombre | Rico | 103.539213 | 176.223837 | 46.228944 | 16721.681442 |
| 446 | Hombre | Pobre | 41.594394 | 197.881419 | 32.933740 | 16207.016934 |
| 447 | Hombre | Pobre | 71.898854 | 145.728011 | 37.305398 | 20894.032053 |
| 448 | Hombre | Clase media | 68.221683 | 154.474879 | 26.519632 | 27312.252693 |
| 455 | Hombre | Pobre | 88.389670 | 172.966093 | 34.949802 | 13803.713538 |
| 456 | Hombre | Pobre | 76.368381 | 157.970215 | 21.940201 | 23526.498047 |
| 457 | Hombre | Clase media | 58.723643 | 133.664295 | 34.027605 | 23433.524772 |
| 459 | Hombre | Clase media | 47.539380 | 165.524175 | 26.547940 | 20599.364920 |
| 462 | Hombre | Pobre | 108.146843 | 172.085807 | 16.258728 | 17816.842928 |
| 465 | Hombre | Clase media | 79.784361 | 109.010498 | 29.763316 | 17210.205430 |
| 466 | Hombre | Clase media | 42.912054 | 176.443106 | 47.247449 | 17251.478254 |
| 468 | Hombre | Pobre | 37.788970 | 158.703511 | 21.134936 | 20197.266842 |
| 469 | Hombre | Rico | 81.108639 | 164.816241 | 44.916621 | 19450.230796 |
| 471 | Hombre | Rico | 49.753602 | 175.617097 | 30.343496 | 15476.742356 |
| 473 | Hombre | Clase media | 68.577303 | 132.877746 | 28.270287 | 23072.652798 |
| 474 | Hombre | Pobre | 46.004514 | 177.100838 | 41.477903 | 16277.318072 |
| 476 | Hombre | Pobre | 48.053292 | 172.373627 | 32.744511 | 18838.871788 |
| 481 | Hombre | Pobre | 29.028383 | 189.397032 | 47.450977 | 23384.297552 |
| 482 | Hombre | Rico | 90.728441 | 153.570766 | 17.133635 | 18595.162047 |
| 483 | Hombre | Rico | 8.476032 | 185.060423 | 61.716743 | 23285.320191 |
| 484 | Hombre | Clase media | 86.395015 | 200.089372 | 40.268949 | 12000.580239 |
| 486 | Hombre | Clase media | 80.325793 | 186.310876 | 9.222875 | 16280.049197 |
| 487 | Hombre | Rico | 103.346903 | 136.358536 | 29.166839 | 14705.324995 |
| 488 | Hombre | Rico | 62.548446 | 164.920059 | 12.892936 | 19948.486048 |

```
491  Hombre            Pobre   97.932800  133.694422  23.530031  15053.198346
492  Hombre            Pobre   84.575073  184.811158  34.008297  14011.828009
494  Hombre             Rico   56.315027  169.029406  26.561931  15756.636840
496  Hombre             Rico   54.667400  170.864904  26.502169  18674.708651
497  Hombre             Rico   68.628059  134.685613  27.459753  22361.539315
498  Hombre      Clase media   20.073167  159.974501  26.578268  18438.432852

[257 rows x 6 columns]
Mujer
      Sexo Estado Economico        Peso      Altura       Edad      Ingresos
4    Mujer      Clase media   61.181933  121.918602  13.601447  23901.455692
5    Mujer      Clase media   64.653648  202.156947  14.276626  22072.023957
6    Mujer             Rico   33.608932  149.200442  19.499976  15909.629024
7    Mujer             Rico   86.882174  176.266911  24.902984  20061.186877
8    Mujer             Rico   93.163432  176.640838  22.585424  22440.940719
9    Mujer      Clase media   46.400017  200.391440  42.941121  18356.377285
10   Mujer            Pobre  -10.591598  164.383082  32.754846  16986.821945
13   Mujer      Clase media   58.006239  149.233307  50.702530  15716.073638
14   Mujer      Clase media   40.584091  186.890421  40.545193  19532.323031
15   Mujer      Clase media   58.376074  161.419205  46.065908   8883.606484
16   Mujer             Rico   31.307101  153.446700  18.829034  22110.670594
17   Mujer             Rico   40.944907  134.548742  18.890701  14543.559477
19   Mujer      Clase media   49.984521  124.648857  38.683582  21873.367663
22   Mujer             Rico   75.237019  101.762481  24.728259  10947.541907
26   Mujer            Pobre   39.406656  185.019067  -1.263022  18275.040730
28   Mujer             Rico  107.653876  175.474834  42.926472  17171.668712
29   Mujer             Rico   82.082029  161.359900  16.809352  16155.491060
30   Mujer            Pobre   49.792379  138.004594  54.549133  14002.537593
35   Mujer      Clase media  113.068925  151.408024  24.790545  14588.801427
38   Mujer            Pobre   44.146773  130.819010  38.053475  16099.272054
39   Mujer             Rico   41.869646  197.515862  42.097703  21657.207445
43   Mujer      Clase media   73.714083  179.204190  38.324433  18116.352179
44   Mujer      Clase media   79.519129  127.801687  11.768651  21049.313755
46   Mujer             Rico   29.211076  167.447367  38.544269  19384.407549
47   Mujer            Pobre   72.723934  153.714213  46.077096  18237.589463
48   Mujer      Clase media   34.190581  106.739274  25.692490  18521.332356
49   Mujer      Clase media   65.159198  140.627270  43.060404  18618.656504
50   Mujer            Pobre   76.306046  178.668915   2.754252  20120.687602
51   Mujer             Rico   13.666734  131.516817  31.524921  17073.394765
53   Mujer             Rico   88.061979  142.096469  33.169495  20735.733463
..     ...              ...         ...         ...        ...           ...
440  Mujer      Clase media   99.731452  124.392941  37.549346  18926.785817
441  Mujer      Clase media  100.175276  167.803278  20.802833  20746.853248
443  Mujer             Rico   93.952518  114.858146   1.974032  14196.975239
444  Mujer             Rico   59.199541  159.346651  34.747025  14991.943446
445  Mujer            Pobre   64.633631  162.921895  12.030290  18690.264318
449  Mujer            Pobre   64.397138  249.001372  29.056784  17067.067374
450  Mujer            Pobre   49.764190  169.281061  23.944155  23545.258029
451  Mujer      Clase media   33.518895  140.315611   2.410050  17851.931866
452  Mujer             Rico   45.976064  148.955804  37.476934  12155.950668
453  Mujer      Clase media   42.668972  173.107395  28.296817  17545.223136
454  Mujer             Rico   86.700874  199.352994  44.115582  13951.983880
458  Mujer      Clase media   70.019016  184.194487  30.955388  22317.994180
460  Mujer             Rico   50.142045  172.648798  35.764421  20492.246108
461  Mujer             Rico   74.561375  110.133710  18.076238  11975.177376
463  Mujer      Clase media   97.708714  163.941644  21.955239  16721.414211
464  Mujer            Pobre  109.477794  168.664001  28.741450  16189.254200
```

```
467  Mujer            Pobre   88.191577  190.574839  26.991766  22009.234525
470  Mujer            Pobre   72.331017  177.284841  16.654716  16937.490203
472  Mujer            Pobre   33.872953  197.178530  33.316359  24088.810742
475  Mujer            Pobre   36.891275  142.793779  28.438586  13206.245954
477  Mujer             Rico   43.690748  171.307994  61.710629  14848.942000
478  Mujer     Clase media   95.152456  161.146638  24.924640  12964.438548
479  Mujer     Clase media   25.233586  157.248114  18.161252  18299.639012
480  Mujer     Clase media   97.983450  177.863890  31.581959  19295.551864
485  Mujer     Clase media   27.559226  125.476923  27.978056  17174.897929
489  Mujer     Clase media   75.076798  208.668954  33.940396  13341.673461
490  Mujer            Pobre  119.141026  182.532712  53.067710  14771.470981
493  Mujer             Rico  102.140518  102.495762  31.202585  17172.427267
495  Mujer             Rico   41.865146  137.059163  20.238401  21707.185946
499  Mujer            Pobre   32.392374  159.457208  37.390841  20971.174066

[243 rows x 6 columns]
```

In [51]:
```
# Revisar por ejemplo a las mujeres
agrupados_sexo.get_group('Mujer').head()
```

Out[51]:

| | Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|---|
| **4** | Mujer | Clase media | 61.181933 | 121.918602 | 13.601447 | 23901.455692 |
| **5** | Mujer | Clase media | 64.653648 | 202.156947 | 14.276626 | 22072.023957 |
| **6** | Mujer | Rico | 33.608932 | 149.200442 | 19.499976 | 15909.629024 |
| **7** | Mujer | Rico | 86.882174 | 176.266911 | 24.902984 | 20061.186877 |
| **8** | Mujer | Rico | 93.163432 | 176.640838 | 22.585424 | 22440.940719 |

In [52]:
```
# Agrupemos por más de una categoría

doble_agrupacion = data_3.groupby(['Sexo', 'Estado Economico'])
len(doble_agrupacion)
```

Out[52]: 6

In [53]:
```
# Revisemos los datos agrupados

for nombre, grupo in doble_agrupacion:
    print(nombre)
    print(grupo)
```

```
('Hombre', 'Clase media')
```

| | Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|---|
| 1 | Hombre | Clase media | 113.361591 | 141.806900 | 13.956049 | 15429.476751 |
| 3 | Hombre | Clase media | 18.778331 | 140.283566 | 42.871327 | 17759.165610 |
| 11 | Hombre | Clase media | 75.303962 | 154.932648 | 34.281199 | 19046.369405 |
| 12 | Hombre | Clase media | 62.728649 | 166.651245 | 29.479539 | 17420.128140 |
| 18 | Hombre | Clase media | 84.737532 | 182.552952 | 38.527954 | 20331.740509 |
| 20 | Hombre | Clase media | 71.459400 | 145.489352 | 25.602147 | 22406.519566 |
| 23 | Hombre | Clase media | 79.984957 | 171.061312 | 30.883470 | 21461.458889 |
| 25 | Hombre | Clase media | 62.429489 | 104.401969 | 41.774141 | 14537.604535 |
| 27 | Hombre | Clase media | 58.603480 | 208.998805 | 28.943294 | 24312.776440 |
| 31 | Hombre | Clase media | 72.877127 | 159.872874 | 36.165485 | 18896.238479 |
| 33 | Hombre | Clase media | 60.016595 | 129.018628 | 23.147194 | 20817.312735 |
| 66 | Hombre | Clase media | 64.350584 | 128.344812 | 7.426064 | 20741.219434 |
| 70 | Hombre | Clase media | 47.886301 | 196.167892 | 42.330173 | 19824.013403 |
| 80 | Hombre | Clase media | 57.048224 | 219.346323 | 23.717325 | 19562.343124 |
| 94 | Hombre | Clase media | 8.966734 | 80.530788 | 23.367649 | 15133.907726 |
| 98 | Hombre | Clase media | 47.185271 | 199.193409 | 24.530735 | 8909.287626 |
| 99 | Hombre | Clase media | 71.476700 | 200.842925 | 31.454117 | 20160.135655 |
| 105 | Hombre | Clase media | 32.225120 | 152.711303 | 27.918561 | 18364.197371 |
| 124 | Hombre | Clase media | 74.761507 | 204.088213 | 49.406206 | 11597.170994 |
| 129 | Hombre | Clase media | 67.281228 | 169.837196 | 28.169565 | 16265.567708 |
| 133 | Hombre | Clase media | 89.181271 | 142.470584 | 30.704459 | 22017.648418 |
| 142 | Hombre | Clase media | 28.292559 | 173.433782 | 24.778627 | 15895.079742 |
| 145 | Hombre | Clase media | 61.410065 | 197.841557 | 31.118750 | 17320.943796 |
| 152 | Hombre | Clase media | 53.749082 | 168.421335 | 22.101205 | 17781.669488 |
| 154 | Hombre | Clase media | 72.023968 | 162.920286 | 18.457637 | 14033.748332 |
| 156 | Hombre | Clase media | 65.336157 | 155.785436 | 35.479325 | 21987.594938 |
| 160 | Hombre | Clase media | 71.144149 | 132.204203 | 12.764324 | 19102.775778 |
| 161 | Hombre | Clase media | 37.942915 | 180.842012 | 17.680896 | 20779.873425 |
| 164 | Hombre | Clase media | 80.652286 | 145.304557 | 26.188991 | 19192.399928 |
| 169 | Hombre | Clase media | 76.205452 | 153.226189 | 17.707988 | 22512.223437 |
| .. | ... | ... | ... | ... | ... | ... |
| 287 | Hombre | Clase media | 88.748308 | 140.149895 | 29.236052 | 20365.924913 |
| 308 | Hombre | Clase media | 81.219473 | 149.104467 | 19.677962 | 19565.458058 |
| 314 | Hombre | Clase media | 49.450741 | 143.969513 | 22.803757 | 21101.770904 |
| 319 | Hombre | Clase media | 9.147273 | 158.091097 | 30.191266 | 18735.870473 |
| 322 | Hombre | Clase media | 51.968495 | 161.703072 | 20.344285 | 22535.798177 |
| 323 | Hombre | Clase media | 90.864908 | 142.916965 | 23.535837 | 25300.872208 |
| 326 | Hombre | Clase media | 67.977048 | 132.207643 | 32.298729 | 15487.221855 |
| 327 | Hombre | Clase media | 55.051324 | 151.487822 | 51.491398 | 11012.318715 |
| 347 | Hombre | Clase media | 71.992141 | 121.685631 | 40.952213 | 19893.233672 |
| 359 | Hombre | Clase media | 58.944362 | 199.319895 | 37.835658 | 23116.424249 |
| 360 | Hombre | Clase media | 70.616643 | 191.267992 | 30.131206 | 17096.895779 |
| 366 | Hombre | Clase media | 52.693533 | 160.531939 | 30.533238 | 19534.868425 |
| 385 | Hombre | Clase media | 42.880893 | 180.775584 | 45.527138 | 18664.358558 |
| 387 | Hombre | Clase media | -5.373007 | 176.380157 | 30.847368 | 16180.989149 |
| 391 | Hombre | Clase media | 98.191714 | 182.413066 | 1.615836 | 24189.954460 |
| 400 | Hombre | Clase media | 81.951277 | 144.534233 | 18.555595 | 15909.665281 |
| 406 | Hombre | Clase media | 89.684805 | 114.757779 | 24.998551 | 17398.578530 |
| 422 | Hombre | Clase media | 67.315244 | 136.296781 | 36.193550 | 18140.892134 |
| 424 | Hombre | Clase media | 89.240180 | 164.426038 | 55.068285 | 23698.578183 |
| 429 | Hombre | Clase media | 102.189079 | 181.429941 | 22.637755 | 12896.111635 |
| 434 | Hombre | Clase media | 42.981332 | 130.723606 | 30.946256 | 25408.387951 |
| 448 | Hombre | Clase media | 68.221683 | 154.474879 | 26.519632 | 27312.252693 |
| 457 | Hombre | Clase media | 58.723643 | 133.664295 | 34.027605 | 23433.524772 |
| 459 | Hombre | Clase media | 47.539380 | 165.524175 | 26.547940 | 20599.364920 |

```
465  Hombre      Clase media   79.784361  109.010498  29.763316  17210.205430
466  Hombre      Clase media   42.912054  176.443106  47.247449  17251.478254
473  Hombre      Clase media   68.577303  132.877746  28.270287  23072.652798
484  Hombre      Clase media   86.395015  200.089372  40.268949  12000.580239
486  Hombre      Clase media   80.325793  186.310876   9.222875  16280.049197
498  Hombre      Clase media   20.073167  159.974501  26.578268  18438.432852

[79 rows x 6 columns]
('Hombre', 'Pobre')
       Sexo Estado Economico        Peso      Altura       Edad      Ingresos
2    Hombre            Pobre   61.085082  188.536748  23.820786  13117.313909
40   Hombre            Pobre   77.669863   98.649087  12.787209  17043.917381
41   Hombre            Pobre   69.902331  185.231122  23.337743  14492.387956
42   Hombre            Pobre   56.935984  192.875052  10.721323  18044.052640
55   Hombre            Pobre   80.401239  125.134461  43.219089  19964.814527
58   Hombre            Pobre   72.370506  186.351808  31.013231  18443.760802
62   Hombre            Pobre   82.419144  174.803364  16.645850  11925.884175
69   Hombre            Pobre   70.757442  154.480758   1.709324  17079.587651
71   Hombre            Pobre   87.442687  132.773588   5.099336  16298.200814
73   Hombre            Pobre   42.533171  178.977269  21.317267  26278.260626
76   Hombre            Pobre   37.319988  176.070722  16.651778  22445.772691
82   Hombre            Pobre   76.512764  177.551124  25.448195  20954.493649
84   Hombre            Pobre   66.055105  182.142501  28.101791  13953.708303
85   Hombre            Pobre   56.185612  140.270492  34.417336  17906.470182
92   Hombre            Pobre  107.116220  125.632428  31.992832  20915.430153
95   Hombre            Pobre   33.433925  162.257569  31.687011  17540.187899
102  Hombre            Pobre   70.903128  149.995923  21.932181  14234.073705
109  Hombre            Pobre   28.737584  149.920924  29.940454  14305.137371
122  Hombre            Pobre   65.116991  163.406115  14.057269  16741.430195
125  Hombre            Pobre   64.251038  173.279617  26.199963  17501.476751
132  Hombre            Pobre   66.967745  143.426572  41.531643  16352.819683
140  Hombre            Pobre   81.031157  177.495365  41.780760  14937.911594
173  Hombre            Pobre   53.358389  160.238799  35.973861  23711.590006
182  Hombre            Pobre   59.556525  152.899703  23.486796  19427.459654
187  Hombre            Pobre   87.943293  170.310458  44.367865  20185.988784
192  Hombre            Pobre   45.822645  208.196153  22.984887  10325.578441
195  Hombre            Pobre   87.792280  197.831857  39.269062  18269.166510
197  Hombre            Pobre   79.201295  120.028044  53.335600  17146.715584
198  Hombre            Pobre   82.644864  130.847526  35.936828  20922.353342
200  Hombre            Pobre   50.244830  156.895947  39.538906  16306.172838
..      ...              ...         ...         ...        ...           ...
361  Hombre            Pobre   59.721446  195.300500  20.889848  13772.671426
371  Hombre            Pobre   63.974942  135.769918  45.714749  14409.635481
372  Hombre            Pobre   66.770001  129.397401  25.189691  20270.104056
377  Hombre            Pobre   43.439737  176.989470  42.170180  10211.492785
380  Hombre            Pobre   -0.594288  199.450061  40.133187  13637.259992
381  Hombre            Pobre   52.240939  176.529729  28.708246  14266.177331
386  Hombre            Pobre   36.147466  124.448146  18.230693  10172.365944
396  Hombre            Pobre   10.236110  108.413935  50.891757  19176.227971
397  Hombre            Pobre   95.835724  143.780147  27.516728  16250.348906
403  Hombre            Pobre   36.064231  119.758987  40.155327  24448.713357
407  Hombre            Pobre   65.940690  167.834534  30.420582  21660.033196
408  Hombre            Pobre   43.827918  173.436901  37.223913  18440.188485
410  Hombre            Pobre   57.754743  238.765490  32.645179  14054.745414
412  Hombre            Pobre   95.145725  137.006191  38.568241  10620.904879
419  Hombre            Pobre   82.893335  137.083879  45.023777  20039.627346
423  Hombre            Pobre   74.161769  104.032057  34.285255  17483.285051
```

```
425  Hombre            Pobre   62.442591  149.205524  23.412649  21823.116103
430  Hombre            Pobre   89.740363  127.463811  37.137120  20859.814350
437  Hombre            Pobre   69.696650  181.710060  32.928282  19270.545601
446  Hombre            Pobre   41.594394  197.881419  32.933740  16207.016934
447  Hombre            Pobre   71.898854  145.728011  37.305398  20894.032053
455  Hombre            Pobre   88.389670  172.966093  34.949802  13803.713538
456  Hombre            Pobre   76.368381  157.970215  21.940201  23526.498047
462  Hombre            Pobre  108.146843  172.085807  16.258728  17816.842928
468  Hombre            Pobre   37.788970  158.703511  21.134936  20197.266842
474  Hombre            Pobre   46.004514  177.100838  41.477903  16277.318072
476  Hombre            Pobre   48.053292  172.373627  32.744511  18838.871788
481  Hombre            Pobre   29.028383  189.397032  47.450977  23384.297552
491  Hombre            Pobre   97.932800  133.694422  23.530031  15053.198346
492  Hombre            Pobre   84.575073  184.811158  34.008297  14011.828009

[92 rows x 6 columns]
('Hombre', 'Rico')
       Sexo Estado Economico        Peso      Altura       Edad      Ingresos
0    Hombre             Rico   33.346201  178.854874  45.385946  20813.470719
21   Hombre             Rico   48.845505  171.255861  36.341825  17866.847612
24   Hombre             Rico   69.516723  161.285463  14.448105  16987.956652
32   Hombre             Rico   60.457446  108.682354  32.892411  18915.110284
34   Hombre             Rico   56.810920  131.873287  53.618540  14463.387698
36   Hombre             Rico   86.511228  173.451385  30.681507  24208.054556
37   Hombre             Rico   65.079106  130.523457  31.906173  16176.374604
45   Hombre             Rico   98.355855  177.743133  42.185538  17108.076404
52   Hombre             Rico   16.357960  161.415063  23.518715  17956.664940
54   Hombre             Rico   90.269989  164.009535  40.801611  19810.412952
61   Hombre             Rico  119.871572  139.603747  17.506358  21872.829970
65   Hombre             Rico   91.592110  207.311949  43.121437  22271.013418
67   Hombre             Rico  116.414782  187.965914  26.974854  21512.618242
83   Hombre             Rico   83.954548  181.313569  20.586196  21407.691451
87   Hombre             Rico   84.336987  143.520413  27.696125  21248.373752
88   Hombre             Rico   71.976589  139.786804  34.705702  24534.843353
97   Hombre             Rico   50.614080  164.790056  29.132660  19756.732213
101  Hombre             Rico   46.458550  145.552165  26.427287  17600.660775
112  Hombre             Rico   58.028707  201.535258  33.085161  23547.997045
128  Hombre             Rico   43.930688  158.616771  51.235194  19351.295016
137  Hombre             Rico   24.057271  167.834114  28.061627  13979.456075
143  Hombre             Rico   73.166341  139.562934  46.144969  18209.886378
144  Hombre             Rico   50.759390  188.077390  46.749676  17110.042990
146  Hombre             Rico   92.844424  154.015235  33.895451  21374.180733
158  Hombre             Rico   48.379885  177.556822  35.051575  17185.182022
176  Hombre             Rico   78.243839  120.968502  38.849695  20598.864745
194  Hombre             Rico   59.628517  221.169610  28.883151  16531.852308
196  Hombre             Rico   68.474471  136.375295  25.028265  10451.648858
202  Hombre             Rico   52.015486  108.919569  34.883346  21533.348317
208  Hombre             Rico   40.210148  164.197095  52.882622  12108.463127
..      ...              ...         ...         ...        ...           ...
370  Hombre             Rico   60.990121  145.784753  43.416678  18951.725008
374  Hombre             Rico   12.467165  175.911061  24.315441  19070.713085
375  Hombre             Rico   73.547270  187.090573  24.154273  22561.767189
376  Hombre             Rico   66.418609  151.011497   9.401122  22299.116326
379  Hombre             Rico   35.335811   98.880741  40.228614  12615.221418
382  Hombre             Rico   97.878594  209.525001  24.580961  11753.881638
383  Hombre             Rico   83.377093  142.466191  31.729866  14508.470685
389  Hombre             Rico   66.066553  177.017183  38.720558  18858.171229
```

```
390  Hombre            Rico   85.919046  178.720784  46.043863  17262.071495
392  Hombre            Rico   65.279682  144.962402  37.032584  17292.637167
394  Hombre            Rico    6.244182  114.856046  23.071098  16436.944615
395  Hombre            Rico  113.731133  166.918998  55.689046  15169.863621
402  Hombre            Rico   68.820446  194.641838  32.837234  14146.996708
404  Hombre            Rico   54.466563  259.014456  22.598526  20062.349218
414  Hombre            Rico   61.059092  185.772039  28.812479  21312.380597
416  Hombre            Rico   22.317768  121.992236  42.344332  21257.369922
426  Hombre            Rico   83.217765  124.498500  33.173307  13472.989111
427  Hombre            Rico   47.336768  163.141672  25.802270  19297.591520
435  Hombre            Rico   93.497424  131.653485  18.024017  21811.938090
439  Hombre            Rico   29.188015  151.794286  23.554060  21776.334150
442  Hombre            Rico  103.539213  176.223837  46.228944  16721.681442
469  Hombre            Rico   81.108639  164.816241  44.916621  19450.230796
471  Hombre            Rico   49.753602  175.617097  30.343496  15476.742356
482  Hombre            Rico   90.728441  153.570766  17.133635  18595.162047
483  Hombre            Rico    8.476032  185.060423  61.716743  23285.320191
487  Hombre            Rico  103.346903  136.358536  29.166839  14705.324995
488  Hombre            Rico   62.548446  164.920059  12.892936  19948.486048
494  Hombre            Rico   56.315027  169.029406  26.561931  15756.636840
496  Hombre            Rico   54.667400  170.864904  26.502169  18674.708651
497  Hombre            Rico   68.628059  134.685613  27.459753  22361.539315

[86 rows x 6 columns]
('Mujer', 'Clase media')
      Sexo Estado Economico        Peso      Altura       Edad      Ingresos
4    Mujer      Clase media   61.181933  121.918602  13.601447  23901.455692
5    Mujer      Clase media   64.653648  202.156947  14.276626  22072.023957
9    Mujer      Clase media   46.400017  200.391440  42.941121  18356.377285
13   Mujer      Clase media   58.006239  149.233307  50.702530  15716.073638
14   Mujer      Clase media   40.584091  186.890421  40.545193  19532.323031
15   Mujer      Clase media   58.376074  161.419205  46.065908   8883.606484
19   Mujer      Clase media   49.984521  124.648857  38.683582  21873.367663
35   Mujer      Clase media  113.068925  151.408024  24.790545  14588.801427
43   Mujer      Clase media   73.714083  179.204190  38.324433  18116.352179
44   Mujer      Clase media   79.519129  127.801687  11.768651  21049.313755
48   Mujer      Clase media   34.190581  106.739274  25.692490  18521.332356
49   Mujer      Clase media   65.159198  140.627270  43.060404  18618.656504
56   Mujer      Clase media   45.364629  173.188047  32.123878  26850.031568
64   Mujer      Clase media   98.127548   97.993416  41.071870  16587.400568
75   Mujer      Clase media   61.935919  127.210796  39.480257  12950.379215
78   Mujer      Clase media  103.752724  172.713283  29.421074  17410.745220
93   Mujer      Clase media   64.142693  182.284956  20.152621  19535.513508
96   Mujer      Clase media   67.088138  178.164939  30.300313  21062.260116
100  Mujer      Clase media   69.128470  110.085675  41.432815  21777.022006
114  Mujer      Clase media   84.935086  144.519525  15.224136  12965.931147
115  Mujer      Clase media   71.699327  138.291325  43.926612   8867.894779
120  Mujer      Clase media   82.914338  140.542303  16.436442  16210.734443
130  Mujer      Clase media  105.015537  175.033642  33.122502  13890.115979
131  Mujer      Clase media   40.435816  184.394914  48.267761  27458.032466
138  Mujer      Clase media  130.575444  155.250332  29.936659  14617.028658
139  Mujer      Clase media   97.930520  179.434862  46.500604  19071.790553
147  Mujer      Clase media   69.487956  148.540740   7.564566  17249.554327
162  Mujer      Clase media   25.338716  133.880423  11.115671  22896.393558
168  Mujer      Clase media   81.486771  171.591400  32.257755  15261.740626
170  Mujer      Clase media   59.593067  111.613783  48.476343  16390.113744
..     ...              ...         ...         ...         ...           ...
```

```
291  Mujer      Clase media   85.008406  145.444154  38.334064  19347.815824
292  Mujer      Clase media   94.019245  134.350068  46.648633  13670.372656
295  Mujer      Clase media   73.199613  101.294508  20.045955  16901.596893
299  Mujer      Clase media   32.295227  139.460676  25.564197  22687.438810
324  Mujer      Clase media   17.742725  200.695972  24.428239  23429.755194
330  Mujer      Clase media   30.220156  188.649133  28.094260  15818.383084
336  Mujer      Clase media   93.890000  184.234879  29.807593  17174.265518
338  Mujer      Clase media  113.385329  151.312295  18.044763  12155.339938
352  Mujer      Clase media   51.534470  145.830080  25.199894  20505.413871
355  Mujer      Clase media    4.043933  157.121423  37.843929  23832.230627
364  Mujer      Clase media   60.656545  108.589947  13.200165  22114.201757
365  Mujer      Clase media   35.191447  100.928075  47.861974  18285.758426
393  Mujer      Clase media   61.446876  194.486431  43.421100  16679.748892
399  Mujer      Clase media   37.445672  119.887149  34.798536  21728.827447
405  Mujer      Clase media   40.947007  147.064771  32.093009  18760.792263
411  Mujer      Clase media   75.054157  134.985146  23.549993  16699.311266
420  Mujer      Clase media    2.971024  198.889518  31.823646  19281.399606
433  Mujer      Clase media   90.072923  143.939242  31.986288  16445.277829
436  Mujer      Clase media   88.200555  154.997915  56.282319  18470.944226
440  Mujer      Clase media   99.731452  124.392941  37.549346  18926.785817
441  Mujer      Clase media  100.175276  167.803278  20.802833  20746.853248
451  Mujer      Clase media   33.518895  140.315611   2.410050  17851.931866
453  Mujer      Clase media   42.668972  173.107395  28.296817  17545.223136
458  Mujer      Clase media   70.019016  184.194487  30.955388  22317.994180
463  Mujer      Clase media   97.708714  163.941644  21.955239  16721.414211
478  Mujer      Clase media   95.152456  161.146638  24.924640  12964.438548
479  Mujer      Clase media   25.233586  157.248114  18.161252  18299.639012
480  Mujer      Clase media   97.983450  177.863890  31.581959  19295.551864
485  Mujer      Clase media   27.559226  125.476923  27.978056  17174.897929
489  Mujer      Clase media   75.076798  208.668954  33.940396  13341.673461

[81 rows x 6 columns]
('Mujer', 'Pobre')
      Sexo Estado Economico        Peso      Altura       Edad      Ingresos
10   Mujer            Pobre  -10.591598  164.383082  32.754846  16986.821945
26   Mujer            Pobre   39.406656  185.019067  -1.263022  18275.040730
30   Mujer            Pobre   49.792379  138.004594  54.549133  14002.537593
38   Mujer            Pobre   44.146773  130.819010  38.053475  16099.272054
47   Mujer            Pobre   72.723934  153.714213  46.077096  18237.589463
50   Mujer            Pobre   76.306046  178.668915   2.754252  20120.687602
57   Mujer            Pobre   60.671523  188.907236  21.466557  18478.147033
68   Mujer            Pobre   67.675218  146.485817  64.969351  14187.588761
79   Mujer            Pobre   67.102038  183.796384  19.839221  16390.748120
81   Mujer            Pobre   73.132096  173.157547  18.556288  17312.467989
86   Mujer            Pobre   91.403481  191.307139  50.406562  21086.182126
89   Mujer            Pobre   49.646347  180.556228  17.887297  11225.830889
90   Mujer            Pobre   57.777721  151.619749  34.111365  18354.466639
91   Mujer            Pobre   55.742644  135.803350  15.772402  15436.605684
108  Mujer            Pobre   62.114820  131.517826  38.310871  15349.566255
110  Mujer            Pobre   71.887446  198.862489  20.647546  15948.680061
113  Mujer            Pobre  103.361196  195.142596  38.342012  17138.131235
119  Mujer            Pobre   94.590317  168.283630  30.226372  16066.098630
123  Mujer            Pobre  109.520847  136.027557  24.912961  22143.892592
134  Mujer            Pobre   74.481450  121.425482  12.092921  15359.534872
148  Mujer            Pobre   17.796603   95.094290  30.708652  16173.731107
151  Mujer            Pobre   75.925932  175.197635  16.825644  23149.792189
153  Mujer            Pobre   70.568414  147.140718  33.488144  20937.014220
```

```
155  Mujer            Pobre   31.786478  160.203282  29.597544  18782.679045
165  Mujer            Pobre   98.274208  190.128148  27.672008  19425.572798
166  Mujer            Pobre  114.648416  170.062984  32.773839  18605.464690
167  Mujer            Pobre   72.554366  172.525115  31.724301  19176.305684
172  Mujer            Pobre   57.997615  152.128856   9.594383  14117.488612
175  Mujer            Pobre   66.712121  160.602845  28.864015  12479.717766
181  Mujer            Pobre   43.944735  235.766951  35.267650  18677.880788
..     ...              ...         ...         ...        ...           ...
302  Mujer            Pobre   30.776725  134.271993  31.757981  17859.706356
304  Mujer            Pobre   20.108745  169.877812  36.629530  18146.761216
307  Mujer            Pobre  116.380398  185.737326  38.881247  13267.640852
312  Mujer            Pobre   74.154441  131.784818  33.347065  25294.664929
313  Mujer            Pobre   44.979655  166.705233  19.526992  22092.057808
331  Mujer            Pobre  107.687219  182.559672  35.416117  23597.034636
340  Mujer            Pobre   45.164912  151.071697  26.818868  24482.602061
346  Mujer            Pobre   57.030863  190.312031  38.961817  20116.748299
350  Mujer            Pobre   34.407265  141.618760  61.726136  16561.227174
351  Mujer            Pobre   96.894760  126.818514  23.008546  17178.331404
357  Mujer            Pobre   89.065085  165.223819  29.161048  15654.498717
362  Mujer            Pobre   28.321485  185.642064  12.490321  19375.575669
378  Mujer            Pobre   58.232071  169.290022  38.206853  19410.948416
388  Mujer            Pobre   51.883060  160.213965  41.910733  23496.387514
401  Mujer            Pobre   76.288865  180.731817   9.291078  20232.583975
415  Mujer            Pobre   53.583740  195.956124  35.039425  19477.961940
417  Mujer            Pobre   41.845083  200.927158  26.287957  15435.433191
418  Mujer            Pobre  131.582906  214.067832  29.114408  22642.499332
431  Mujer            Pobre   71.324358  170.236415  24.868960  18308.660041
438  Mujer            Pobre  139.203832  188.501848  22.332462  20642.608067
445  Mujer            Pobre   64.633631  162.921895  12.030290  18690.264318
449  Mujer            Pobre   64.397138  249.001372  29.056784  17067.067374
450  Mujer            Pobre   49.764190  169.281061  23.944155  23545.258029
464  Mujer            Pobre  109.477794  168.664001  28.741450  16189.254200
467  Mujer            Pobre   88.191577  190.574839  26.991766  22009.234525
470  Mujer            Pobre   72.331017  177.284841  16.654716  16937.490203
472  Mujer            Pobre   33.872953  197.178530  33.316359  24088.810742
475  Mujer            Pobre   36.891275  142.793779  28.438586  13206.245954
490  Mujer            Pobre  119.141026  182.532712  53.067710  14771.470981
499  Mujer            Pobre   32.392374  159.457208  37.390841  20971.174066

[72 rows x 6 columns]
('Mujer', 'Rico')
      Sexo Estado Economico        Peso      Altura       Edad      Ingresos
6    Mujer             Rico   33.608932  149.200442  19.499976  15909.629024
7    Mujer             Rico   86.882174  176.266911  24.902984  20061.186877
8    Mujer             Rico   93.163432  176.640838  22.585424  22440.940719
16   Mujer             Rico   31.307101  153.446700  18.829034  22110.670594
17   Mujer             Rico   40.944907  134.548742  18.890701  14543.559477
22   Mujer             Rico   75.237019  101.762481  24.728259  10947.541907
28   Mujer             Rico  107.653876  175.474834  42.926472  17171.668712
29   Mujer             Rico   82.082029  161.359900  16.809352  16155.491060
39   Mujer             Rico   41.869646  197.515862  42.097703  21657.207445
46   Mujer             Rico   29.211076  167.447367  38.544269  19384.407549
51   Mujer             Rico   13.666734  131.516817  31.524921  17073.394765
53   Mujer             Rico   88.061979  142.096469  33.169495  20735.733463
59   Mujer             Rico   38.547975  207.020269   9.969273   9722.436525
60   Mujer             Rico   50.727924  142.963689  29.801149  14702.150243
63   Mujer             Rico   68.625278  184.021786  14.924959  15452.909141
```

| | | | | | | |
|---|---|---|---|---|---|---|
| 72 | Mujer | Rico | 17.611407 | 136.276460 | 39.151419 | 14323.478741 |
| 74 | Mujer | Rico | 46.572765 | 104.903638 | 32.986976 | 18970.547004 |
| 77 | Mujer | Rico | 101.703429 | 162.607418 | 16.496269 | 25289.762143 |
| 103 | Mujer | Rico | 17.837618 | 164.985492 | 38.589024 | 20144.321696 |
| 104 | Mujer | Rico | 63.712303 | 118.309269 | 42.510729 | 13758.034751 |
| 106 | Mujer | Rico | 53.087430 | 156.916570 | 47.908183 | 16421.371433 |
| 107 | Mujer | Rico | 50.913768 | 143.597978 | 40.674485 | 16462.990536 |
| 111 | Mujer | Rico | 32.258007 | 181.982122 | 29.797028 | 13971.303493 |
| 116 | Mujer | Rico | 61.537137 | 146.049930 | 44.545353 | 15664.961388 |
| 117 | Mujer | Rico | 57.631477 | 122.243479 | 33.176179 | 21160.182441 |
| 118 | Mujer | Rico | 82.469598 | 131.582969 | -3.962558 | 24931.742458 |
| 121 | Mujer | Rico | 91.615903 | 137.385965 | 6.142971 | 25276.314099 |
| 126 | Mujer | Rico | 83.887658 | 178.816849 | 6.637029 | 18798.361813 |
| 127 | Mujer | Rico | 79.304307 | 191.432436 | 36.514161 | 18456.447229 |
| 135 | Mujer | Rico | 53.411018 | 112.259004 | 36.701914 | 24274.772720 |
| .. | ... | ... | ... | ... | ... | ... |
| 317 | Mujer | Rico | 26.833444 | 105.435801 | 47.206224 | 21848.467222 |
| 320 | Mujer | Rico | 83.081458 | 181.869307 | 43.670882 | 17490.104736 |
| 321 | Mujer | Rico | 73.198535 | 130.524214 | 34.623432 | 23210.756383 |
| 328 | Mujer | Rico | 39.032603 | 133.837154 | 29.088720 | 16872.422268 |
| 329 | Mujer | Rico | 102.637227 | 182.814687 | 37.744539 | 17131.033159 |
| 335 | Mujer | Rico | 68.375668 | 195.219389 | 29.080453 | 16587.816787 |
| 339 | Mujer | Rico | 41.027510 | 151.082450 | 12.017843 | 15493.777919 |
| 342 | Mujer | Rico | 79.749131 | 149.294771 | 32.529454 | 19491.172735 |
| 344 | Mujer | Rico | 113.286186 | 207.576025 | 28.371911 | 16571.799871 |
| 345 | Mujer | Rico | 60.543742 | 129.409801 | 37.710709 | 15808.173674 |
| 349 | Mujer | Rico | 75.865513 | 50.767222 | 34.020550 | 16220.874223 |
| 353 | Mujer | Rico | 83.715826 | 139.801328 | 26.940822 | 17754.396893 |
| 356 | Mujer | Rico | 66.576031 | 140.543020 | 23.445791 | 20786.779439 |
| 373 | Mujer | Rico | 47.247790 | 186.086650 | 38.339240 | 14677.254733 |
| 384 | Mujer | Rico | 26.127955 | 206.462144 | 28.998159 | 17618.740882 |
| 398 | Mujer | Rico | 57.986185 | 157.145931 | 15.488776 | 16676.766602 |
| 409 | Mujer | Rico | 75.624786 | 135.965823 | 37.072278 | 19643.919862 |
| 413 | Mujer | Rico | 82.387699 | 178.722721 | 17.650421 | 18727.713406 |
| 421 | Mujer | Rico | 61.428313 | 110.925679 | 11.079925 | 15396.424417 |
| 428 | Mujer | Rico | 35.376707 | 133.171485 | 43.384477 | 14627.529236 |
| 432 | Mujer | Rico | 72.209276 | 192.124342 | 12.846644 | 20314.362548 |
| 443 | Mujer | Rico | 93.952518 | 114.858146 | 1.974032 | 14196.975239 |
| 444 | Mujer | Rico | 59.199541 | 159.346651 | 34.747025 | 14991.943446 |
| 452 | Mujer | Rico | 45.976064 | 148.955804 | 37.476934 | 12155.950668 |
| 454 | Mujer | Rico | 86.700874 | 199.352994 | 44.115582 | 13951.983880 |
| 460 | Mujer | Rico | 50.142045 | 172.648798 | 35.764421 | 20492.246108 |
| 461 | Mujer | Rico | 74.561375 | 110.133710 | 18.076238 | 11975.177376 |
| 477 | Mujer | Rico | 43.690748 | 171.307994 | 61.710629 | 14848.942000 |
| 493 | Mujer | Rico | 102.140518 | 102.495762 | 31.202585 | 17172.427267 |
| 495 | Mujer | Rico | 41.865146 | 137.059163 | 20.238401 | 21707.185946 |

[90 rows x 6 columns]

# Operación sobre datos agrupados

In [54]:
```
doble_agrupacion.sum()
```

Out[54]:

| Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|
| **Hombre** | **Clase media** | 5013.312283 | 12436.794040 | 2257.234591 | 1.455356e+06 |
| | **Pobre** | 5741.598002 | 14828.038511 | 2697.589791 | 1.586103e+06 |
| | **Rico** | 5372.297226 | 13440.993632 | 2683.046478 | 1.578362e+06 |
| **Mujer** | **Clase media** | 5236.669807 | 12549.745940 | 2455.612164 | 1.457918e+06 |
| | **Pobre** | 4681.788903 | 12054.681951 | 2132.585335 | 1.310266e+06 |
| | **Rico** | 5598.311570 | 13922.070760 | 2663.986143 | 1.617598e+06 |

In [55]:
```
doble_agrupacion.mean()
```

Out[55]:

| Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|
| **Hombre** | **Clase media** | 63.459649 | 157.427773 | 28.572590 | 18422.226299 |
| | **Pobre** | 62.408674 | 161.174332 | 29.321628 | 17240.245195 |
| | **Rico** | 62.468572 | 156.290624 | 31.198215 | 18353.048110 |
| **Mujer** | **Clase media** | 64.650245 | 154.935135 | 30.316200 | 17998.993417 |
| | **Pobre** | 65.024846 | 167.426138 | 29.619241 | 18198.141683 |
| | **Rico** | 62.203462 | 154.689675 | 29.599846 | 17973.311188 |

In [56]:
```
doble_agrupacion.size()
```

Out[56]:
```
Sexo    Estado Economico
Hombre  Clase media         79
        Pobre               92
        Rico                86
Mujer   Clase media         81
        Pobre               72
        Rico                90
dtype: int64
```

In [57]:
```
doble_agrupacion.describe()
```

Out[57]:

| | | Peso | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | count | mean | std | min | 25% | 50% | 75% |
| Sexo | Estado Economico | | | | | | | |
| Hombre | Clase media | 79.0 | 63.459649 | 23.902090 | -5.373007 | 50.709618 | 67.281228 | 80.155375 |
| | Pobre | 92.0 | 62.408674 | 23.941858 | -0.594288 | 43.730873 | 64.684014 | 78.052721 |
| | Rico | 86.0 | 62.468572 | 26.900411 | -9.920644 | 46.609467 | 61.028891 | 82.690483 |
| Mujer | Clase media | 81.0 | 64.650245 | 28.358615 | 2.971024 | 42.565867 | 64.653648 | 88.200555 |
| | Pobre | 72.0 | 65.024846 | 29.179655 | -10.591598 | 44.771435 | 65.672876 | 77.007572 |
| | Rico | 90.0 | 62.203462 | 25.902806 | 8.305391 | 41.236919 | 61.482725 | 82.449123 |

6 rows × 32 columns

In [58]:
```
doble_agrupacion['Ingresos'].describe()
```

Out[58]:

| | | count | mean | std | min | 25% | 50% |
|---|---|---|---|---|---|---|---|
| Sexo | Estado Economico | | | | | | |
| Hombre | Clase media | 79.0 | 18422.226299 | 3633.789522 | 8909.287626 | 16272.808453 | 18664.35855 |
| | Pobre | 92.0 | 17240.245195 | 3696.776907 | 6099.473523 | 14407.276102 | 17390.48678 |
| | Rico | 86.0 | 18353.048110 | 3440.538566 | 10451.648858 | 15861.784249 | 18581.68290 |
| Mujer | Clase media | 81.0 | 17998.993417 | 3549.832838 | 8867.894779 | 15818.383084 | 17851.93186 |
| | Pobre | 72.0 | 18198.141683 | 3247.672597 | 11225.830889 | 16090.978698 | 18192.17534 |
| | Rico | 90.0 | 17973.311188 | 3530.933151 | 9722.436525 | 15463.126335 | 17671.06341 |

In [59]:
```
doble_agrupacion.aggregate(
    {
        'Ingresos': np.sum,
        'Edad': np.mean,
        'Altura': np.std
    }
)
```

Out[59]:

| | | Ingresos | Edad | Altura |
|---|---|---|---|---|
| **Sexo** | **Estado Economico** | | | |
| **Hombre** | **Clase media** | 1.455356e+06 | 28.572590 | 27.826528 |
| | **Pobre** | 1.586103e+06 | 29.321628 | 26.983848 |
| | **Rico** | 1.578362e+06 | 31.198215 | 29.242582 |
| **Mujer** | **Clase media** | 1.457918e+06 | 30.316200 | 28.905208 |
| | **Pobre** | 1.310266e+06 | 29.619241 | 27.561274 |
| | **Rico** | 1.617598e+06 | 29.599846 | 31.355662 |

In [60]:
```
doble_agrupacion.aggregate(
    {
        'Edad': np.mean,
        'Altura': lambda h: np.mean(h)/np.std(h)
    }
)
```

Out[60]:

| | | Edad | Altura |
|---|---|---|---|
| **Sexo** | **Estado Economico** | | |
| **Hombre** | **Clase media** | 28.572590 | 5.693621 |
| | **Pobre** | 29.321628 | 6.005722 |
| | **Rico** | 31.198215 | 5.375972 |
| **Mujer** | **Clase media** | 30.316200 | 5.393508 |
| | **Pobre** | 29.619241 | 6.117318 |
| | **Rico** | 29.599846 | 4.961027 |

In [61]:
```
doble_agrupacion.aggregate([np.sum, np.mean, np.std])
```

Out[61]:

| | | Peso | | | Altura | | | E |
|---|---|---|---|---|---|---|---|---|
| | | sum | mean | std | sum | mean | std | s |
| Sexo | Estado Economico | | | | | | | |
| Hombre | Clase media | 5013.312283 | 63.459649 | 23.902090 | 12436.794040 | 157.427773 | 27.826528 | 2 |
| | Pobre | 5741.598002 | 62.408674 | 23.941858 | 14828.038511 | 161.174332 | 26.983848 | 2 |
| | Rico | 5372.297226 | 62.468572 | 26.900411 | 13440.993632 | 156.290624 | 29.242582 | 2 |
| Mujer | Clase media | 5236.669807 | 64.650245 | 28.358615 | 12549.745940 | 154.935135 | 28.905208 | 2 |
| | Pobre | 4681.788903 | 65.024846 | 29.179655 | 12054.681951 | 167.426138 | 27.561274 | 2 |
| | Rico | 5598.311570 | 62.203462 | 25.902806 | 13922.070760 | 154.689675 | 31.355662 | 2 |

In [62]:
```
doble_agrupacion.aggregate([lambda x: np.mean(x) / np.std(x)])
```

Out[62]:

| | | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|
| | | <lambda> | <lambda> | <lambda> | <lambda> |
| Sexo | Estado Economico | | | | |
| Hombre | Clase media | 2.671948 | 5.693621 | 2.641132 | 5.102096 |
| | Pobre | 2.620960 | 6.005722 | 2.659501 | 4.689142 |
| | Rico | 2.335836 | 5.375972 | 2.761100 | 5.365640 |
| Mujer | Clase media | 2.293943 | 5.393508 | 2.584400 | 5.101969 |
| | Pobre | 2.244069 | 6.117318 | 2.422572 | 5.642764 |
| | Rico | 2.414871 | 4.961027 | 2.301020 | 5.118761 |

# Transformación de datos

In [63]:
```
## Zscore

zscore = lambda x: (x-x.mean())/x.std() # A cada valore le quito el valor prem
io y divido por su desviación típica.
```

In [64]:
```
doble_agrupacion.transform(zscore).head()
```

Out[64]:

| | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|
| **0** | -1.082599 | 0.771623 | 1.248319 | 0.715127 |
| **1** | 2.087765 | -0.561366 | -1.342514 | -0.823589 |
| **2** | -0.055284 | 1.014029 | -0.496213 | -1.115277 |
| **3** | -1.869348 | -0.616110 | 1.313324 | -0.182471 |
| **4** | -0.122302 | -1.142235 | -1.416079 | 1.662744 |

In [65]:
```
## Con valores perdidos

fill_na_mean = lambda x : x.fillna(x.mean())
doble_agrupacion.transform(fill_na_mean).head()
```

Out[65]:

| | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|
| **0** | 33.346201 | 178.854874 | 45.385946 | 20813.470719 |
| **1** | 113.361591 | 141.806900 | 13.956049 | 15429.476751 |
| **2** | 61.085082 | 188.536748 | 23.820786 | 13117.313909 |
| **3** | 18.778331 | 140.283566 | 42.871327 | 17759.165610 |
| **4** | 61.181933 | 121.918602 | 13.601447 | 23901.455692 |

# Operaciones diversas y útiles

In [66]:
```
## Primera fila de cada grupo
doble_agrupacion.head(1)
```

Out[66]:

| | Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|---|
| **0** | Hombre | Rico | 33.346201 | 178.854874 | 45.385946 | 20813.470719 |
| **1** | Hombre | Clase media | 113.361591 | 141.806900 | 13.956049 | 15429.476751 |
| **2** | Hombre | Pobre | 61.085082 | 188.536748 | 23.820786 | 13117.313909 |
| **4** | Mujer | Clase media | 61.181933 | 121.918602 | 13.601447 | 23901.455692 |
| **6** | Mujer | Rico | 33.608932 | 149.200442 | 19.499976 | 15909.629024 |
| **10** | Mujer | Pobre | -10.591598 | 164.383082 | 32.754846 | 16986.821945 |

In [67]:
```
## Última fila de cada grupo
doble_agrupacion.tail(1)
```

Out[67]:

| | Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|---|
| **489** | Mujer | Clase media | 75.076798 | 208.668954 | 33.940396 | 13341.673461 |
| **492** | Hombre | Pobre | 84.575073 | 184.811158 | 34.008297 | 14011.828009 |
| **495** | Mujer | Rico | 41.865146 | 137.059163 | 20.238401 | 21707.185946 |
| **497** | Hombre | Rico | 68.628059 | 134.685613 | 27.459753 | 22361.539315 |
| **498** | Hombre | Clase media | 20.073167 | 159.974501 | 26.578268 | 18438.432852 |
| **499** | Mujer | Pobre | 32.392374 | 159.457208 | 37.390841 | 20971.174066 |

In [68]:
```
## n-ésima fila de cada grupo
n = 32
doble_agrupacion.nth(n)
```

Out[68]:

| | | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|
| **Sexo** | **Estado Economico** | | | | |
| **Hombre** | **Clase media** | 81.334896 | 208.016617 | 27.813146 | 18749.311183 |
| | **Pobre** | 37.061894 | 147.210041 | 38.761118 | 18044.235768 |
| | **Rico** | 40.995727 | 172.776297 | 30.073019 | 26768.083564 |
| **Mujer** | **Clase media** | 112.710087 | 180.991135 | 14.704474 | 17113.816865 |
| | **Pobre** | 50.861898 | 171.424717 | 29.145658 | 14728.658398 |
| | **Rico** | 21.677604 | 166.956600 | 42.844271 | 20526.553283 |

In [69]:
```
## n-ésima fila de cada grupo
n = 84
doble_agrupacion.nth(n)
```

Out[69]:

| | | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|
| **Sexo** | **Estado Economico** | | | | |
| **Hombre** | **Pobre** | 76.368381 | 157.970215 | 21.940201 | 23526.498047 |
| | **Rico** | 54.667400 | 170.864904 | 26.502169 | 18674.708651 |
| **Mujer** | **Rico** | 86.700874 | 199.352994 | 44.115582 | 13951.983880 |

¿Qué ocurrió con las demás clasificaciones? No se muestran porque existen menos de 84 registros para esas categorías.

In [70]:
```
data_3.head()
```

Out[70]:

| | Sexo | Estado Economico | Peso | Altura | Edad | Ingresos |
|---|---|---|---|---|---|---|
| **0** | Hombre | Rico | 33.346201 | 178.854874 | 45.385946 | 20813.470719 |
| **1** | Hombre | Clase media | 113.361591 | 141.806900 | 13.956049 | 15429.476751 |
| **2** | Hombre | Pobre | 61.085082 | 188.536748 | 23.820786 | 13117.313909 |
| **3** | Hombre | Clase media | 18.778331 | 140.283566 | 42.871327 | 17759.165610 |
| **4** | Mujer | Clase media | 61.181933 | 121.918602 | 13.601447 | 23901.455692 |

# Conjuntos de entrenamiento y testing

Para realizar un modelo predictivo de Machine Learning, un dataset en general se divide en un subconjunto de entrenamiento y otro de testing.

- El conjunto de entrenamiento (generalmente el 75% del dataset original) es para construir el modelo.

```
    - El modelo tiende ajustarse muy bien con respecto a este conjunto de entrena
miento (overfitting)
```

- El conjunto de testing (generalmente el 25% del dataset original) se utiliza para comprobar la eficacia de dicho modelo.

El método más efectivo para crear estos subconjuntos, es elegir de forma aleatoria del dataset original!

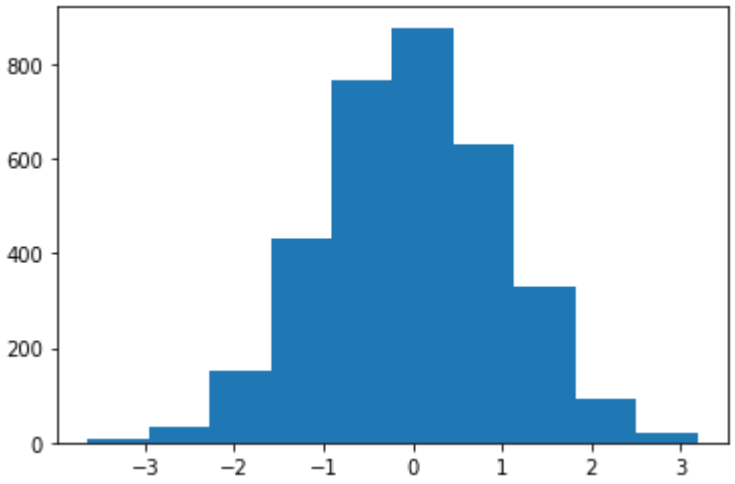
## Dividir utilizando la distribución normal

In [71]:
```
data = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/custome
r-churn-model/Customer Churn Model.txt')
len(data)
```

Out[71]: 3333

In [72]:
```
a = np.random.randn(len(data))
plt.hist(a)
```

Out[72]:
```
(array([  6.,  35., 151., 431., 763., 876., 629., 327.,  94.,  21.]),
 array([-3.63962874, -2.95716081, -2.27469288, -1.59222495, -0.90975702,
        -0.22728909,  0.45517884,  1.13764677,  1.82011469,  2.50258262,
         3.18505055]),
 <a list of 10 Patch objects>)
```

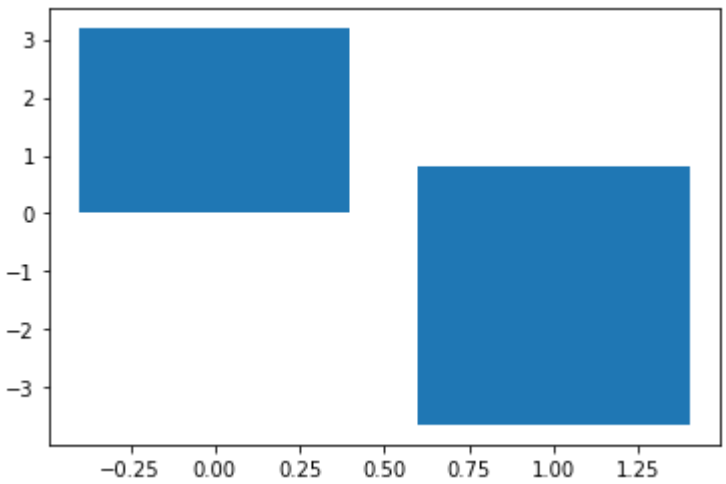



In [73]:
```
## Generamos una condición para obtener un punto de corte entre los conjuntos

check = (a<0.8)
check
```

Out[73]:
```
array([ True,  True, False, ...,  True,  True,  True])
```

In [74]:
```
#Revisemos gráficamente los conjuntos de datos que se pueden generar
plt.bar(check, a)
```

Out[74]:
```
<BarContainer object of 3333 artists>
```



In [75]:
```
# Dividimos la muestra
training = data[check]
testing = data[~check]
print(len(training), len(testing))
```

```
2636 697
```

### Con la librería sklearn

In [76]:
```
from sklearn.model_selection import train_test_split
```

In [77]:
```
# Es muy sencillo con esta librería

train, test = train_test_split(data, test_size=0.2)
```

In [78]:
```
print(len(train), len(test))
```

```
2666 667
```

### Usando una función de shuffle

In [79]:
```
# Desde sklearn
import sklearn
```

In [80]:
```
# Ordenar de forma aleatoria el dataset
data = sklearn.utils.shuffle(data)
```

In [81]:
```
# Hacer la división
pje_corte = int(0.75*len(data))
train_data = data[:pje_corte]
test_data = data[pje_corte+1:]
```

In [82]:
```
print(len(train_data), len(test_data))
```

```
2499 833
```

# Concatenar y apendizar data sets

¿Qué ocurre cuando tienes los datasets separados, con los mismos atributos?

In [83]:
```
red_wine = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/wine/winequality-red.csv', sep=';')
red_wine.head()
```

Out[83]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcoho |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 | 9.4 |
| **1** | 7.8 | 0.88 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.9968 | 3.20 | 0.68 | 9.8 |
| **2** | 7.8 | 0.76 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.9970 | 3.26 | 0.65 | 9.8 |
| **3** | 11.2 | 0.28 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.9980 | 3.16 | 0.58 | 9.8 |
| **4** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 | 9.4 |

In [84]:
```
red_wine.columns.values
```

Out[84]:
```
array(['fixed acidity', 'volatile acidity', 'citric acid',
       'residual sugar', 'chlorides', 'free sulfur dioxide',
       'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol',
       'quality'], dtype=object)
```

In [85]:
```
red_wine.shape
```

Out[85]:
```
(1599, 12)
```

In [86]:
```
white_wine = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/wine/winequality-white.csv', sep=';')
white_wine.head()
```

Out[86]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcoho |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.0 | 0.27 | 0.36 | 20.7 | 0.045 | 45.0 | 170.0 | 1.0010 | 3.00 | 0.45 | 8.8 |
| **1** | 6.3 | 0.30 | 0.34 | 1.6 | 0.049 | 14.0 | 132.0 | 0.9940 | 3.30 | 0.49 | 9.5 |
| **2** | 8.1 | 0.28 | 0.40 | 6.9 | 0.050 | 30.0 | 97.0 | 0.9951 | 3.26 | 0.44 | 10.1 |
| **3** | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 |
| **4** | 7.2 | 0.23 | 0.32 | 8.5 | 0.058 | 47.0 | 186.0 | 0.9956 | 3.19 | 0.40 | 9.9 |

In [87]:
```
white_wine.shape
```

Out[87]:
```
(4898, 12)
```

En python, tenemos dos tipos de ejes:

- axis = 0 denota el eje horizontal (filas)
- axis = 1 denota el eje vertical (columnas)

Para este caso, donde tenemos la misma cantidad de columnas y en el mismo orden, entonces juntaremos ambos datasets de forma horizontal, es decir, agregaremos las filas de ambos datasets

In [88]:
```
## El ´termino es "apilar", donde primero pondremos el data set red_wine y le
 agregaremos el data set  white_wine
wine_data = pd.concat([red_wine, white_wine], axis=0)
wine_data.shape
```

Out[88]: (6497, 12)

In [89]:
```
wine_data.head()
```

Out[89]:

| | fixed acidity | volatile acidity | citric acid | residual sugar | chlorides | free sulfur dioxide | total sulfur dioxide | density | pH | sulphates | alcoho |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 | 9.4 |
| **1** | 7.8 | 0.88 | 0.00 | 2.6 | 0.098 | 25.0 | 67.0 | 0.9968 | 3.20 | 0.68 | 9.8 |
| **2** | 7.8 | 0.76 | 0.04 | 2.3 | 0.092 | 15.0 | 54.0 | 0.9970 | 3.26 | 0.65 | 9.8 |
| **3** | 11.2 | 0.28 | 0.56 | 1.9 | 0.075 | 17.0 | 60.0 | 0.9980 | 3.16 | 0.58 | 9.8 |
| **4** | 7.4 | 0.70 | 0.00 | 1.9 | 0.076 | 11.0 | 34.0 | 0.9978 | 3.51 | 0.56 | 9.4 |

# ¿Qué ocurre cuando tienes cientos datos distribuidos? ¿Cómo se unen?

Este es el caso cuando tenemos registros por día de algún sistema o medición en particular.

En este caso tenemos cerca de 300 bases con respecto a los niveles de contaminación.

In [90]:
```
# Primero revisemos la data que tiene

data = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/distributed-data/001.csv')
data.head()
```

Out[90]:

| | Date | sulfate | nitrate | ID |
|---|---|---|---|---|
| **0** | 2003-01-01 | NaN | NaN | 1 |
| **1** | 2003-01-02 | NaN | NaN | 1 |
| **2** | 2003-01-03 | NaN | NaN | 1 |
| **3** | 2003-01-04 | NaN | NaN | 1 |
| **4** | 2003-01-05 | NaN | NaN | 1 |

In [91]:
```
data.shape
```

Out[91]: (1461, 4)

Procedimiento:

1. Importar el primer fichero (base)
2. Realizar un bucle/ciclo para ir recorriendo todos y cada uno de los ficheros
   - Importante tener una consistencia en el nombre de los ficheros
   - Importamos los ficheros uno a uno
   - Cada uno de ellos debe apendizarse (añadirse al final) del primer fichero que ya habíamos cargado
3. Repetimos el bucle hasta que queden ficheros

In [92]:
```
filepath = '/Users/fsanmartin/python-ml-course-master/datasets/distributed-data/'

data = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/distributed-data/001.csv')

for i in range(2, 333):
    if i < 10:
        filename = '00'+str(i)
    elif i >= 10 and i < 100:
        filename = '0' + str(i)
    else:
        filename = str(i)

    file = filepath + filename + '.csv'

    temp_data = pd.read_csv(file)

    data = pd.concat([data, temp_data], axis=0)
```

In [93]:
```
# Revisemos la información
data.shape
```

Out[93]: (772087, 4)

## Joins de datasets

El caso en que tenemos "n" tablas, donde se unen a través de algún identificador (primary key). Es decir, una base de datos relacional.

En el caso de python, se unen datasets*

In [94]:
```
#Este dataset tiene un detalle con la codificación (generalmente se utiliza utf-8)
data_main = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/athletes/Medals.csv', encoding='ISO-8859-1')
data_main.head()
```

Out[94]:

| | Athlete | Age | Year | Closing Ceremony Date | Gold Medals | Silver Medals | Bronze Medals | Total Medals |
|---|---|---|---|---|---|---|---|---|
| **0** | Michael Phelps | 23.0 | 2008 | 08/24/2008 | 8 | 0 | 0 | 8 |
| **1** | Michael Phelps | 19.0 | 2004 | 08/29/2004 | 6 | 0 | 2 | 8 |
| **2** | Michael Phelps | 27.0 | 2012 | 08/12/2012 | 4 | 2 | 0 | 6 |
| **3** | Natalie Coughlin | 25.0 | 2008 | 08/24/2008 | 1 | 2 | 3 | 6 |
| **4** | Aleksey Nemov | 24.0 | 2000 | 10/01/2000 | 2 | 1 | 3 | 6 |

In [95]:
```
# Revisemos los atletas únicos

a = data_main['Athlete'].unique().tolist()
len(a)
```

Out[95]: 6956

In [96]:
```
data_main.shape
```

Out[96]: (8618, 8)

In [97]:
```
# Obtenemos la base con los países asociados a los atletas

data_country = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/athletes/Athelete_Country_Map.csv', encoding='ISO-8859-1')
```

In [98]:
```
data_country.head()
```

Out[98]:

| | Athlete | Country |
|---|---|---|
| **0** | Michael Phelps | United States |
| **1** | Natalie Coughlin | United States |
| **2** | Aleksey Nemov | Russia |
| **3** | Alicia Coutts | Australia |
| **4** | Missy Franklin | United States |

En este caso, tenemos dos bases con informaciones distintas, pero con los atletas en común.

Revisemos que tengamos la misma cantidad de registros

In [99]:
```
data_country.shape[0]
```

Out[99]: 6970

In [100]:
```
len(a)
```

Out[100]: 6956

¿Por qué existen más registros en la segunda tabla? Quizás puede ser que un mismo atleta olímpico haya jugado para dos países diferentes a lo largo de la historia y además haya ganado medallas. En un caso en concreto, puede ser que un país haya cambiado de nombre (por ejemplo, la división de Yugoslavia en nuevos países)

In [101]:
```
data_country['Athlete'].value_counts().head(10)
```

Out[101]:
```
David Musulbes      2
Vanja Udovicic      2
Gyuzel Manyurova    2
Slobodan Nikic      2
Chen Jing           2
Iván García         2
Dejan Savic         2
Richard Thompson    2
Aleksandar □apic    2
Matt Wells          2
Name: Athlete, dtype: int64
```

Como comprobamos, existen atletas que han participado representando distintos países.

Por último, tenemos una base que asocia el atleta con su deporte

In [102]:
```
data_sports = pd.read_csv('/Users/fsanmartin/python-ml-course-master/datasets/
athletes/Athelete_Sports_Map.csv', encoding='ISO-8859-1')
```

In [103]:
```
data_sports.head()
```

Out[103]:

| | Athlete | Sport |
|---|---|---|
| **0** | Michael Phelps | Swimming |
| **1** | Natalie Coughlin | Swimming |
| **2** | Aleksey Nemov | Gymnastics |
| **3** | Alicia Coutts | Swimming |
| **4** | Missy Franklin | Swimming |

In [104]:
```
data_sports.shape
```

Out[104]:
```
(6975, 2)
```

Aquí también, tenemos más registros. Esto quiere decir que un mismo atleta pudo haber participado a 2 deportes olímpicos.

In [105]:
```
data_sports['Athlete'].value_counts().head(10)
```

Out[105]:
```
Chen Jing           2
Matt Ryan           2
Li Ting             2
Matt Wells          2
Ryan Bailey         2
Jang Seong-Ho       2
Richard Thompson    2
Yang Wei            2
Kim Nam-Sun         2
Nataliya Ivanova    2
Name: Athlete, dtype: int64
```

# ¿Y cómo unimos estos 3 datasets?

En este caso, tenemos la columna Athlete como punto en compun de los data sets. Lo hacemos con "merge"

In [106]:
```
# Separamos el campo de unión
key = 'Athlete'
data_main_country = pd.merge(left = data_main, right = data_country,
                             left_on=key, right_on = key)
```

In [107]:
```
data_main_country.head()
```

Out[107]:

| | Athlete | Age | Year | Closing Ceremony Date | Gold Medals | Silver Medals | Bronze Medals | Total Medals | Country |
|---|---|---|---|---|---|---|---|---|---|
| **0** | Michael Phelps | 23.0 | 2008 | 08/24/2008 | 8 | 0 | 0 | 8 | United States |
| **1** | Michael Phelps | 19.0 | 2004 | 08/29/2004 | 6 | 0 | 2 | 8 | United States |
| **2** | Michael Phelps | 27.0 | 2012 | 08/12/2012 | 4 | 2 | 0 | 6 | United States |
| **3** | Natalie Coughlin | 25.0 | 2008 | 08/24/2008 | 1 | 2 | 3 | 6 | United States |
| **4** | Natalie Coughlin | 21.0 | 2004 | 08/29/2004 | 2 | 2 | 1 | 5 | United States |

¿Y qué ocurrió con la dimensión del dataframe? Lo que se utilizó, es una técnica "innerjoin" donde se hace la combinación de ambos datasets, por lo tanto si un mismo atleta se relaciona con más de un país, entonces se crearán registros para cada país con los mismos resultados obtenidos en el mismo año.

In [108]:
```
data_main_country.shape
```

Out[108]: (8657, 9)

¿Qué hacemos entonces? Se recomienda tener una relación uno a uno, en este caso, cada atleta se relacione con un solo país. Por lo tanto se procederá a eliminar los registros duplicados de la base que relaciona el atleta con su país.

In [109]:
```
data_country_dp = data_country.drop_duplicates(subset='Athlete')
```

In [110]:
```
data_main_country_dp = pd.merge(left = data_main, right = data_country_dp,
                                left_on=key, right_on = key)
```

In [111]:
```
data_main_country_dp.shape
```

Out[111]: (8618, 9)

De esta forma corregimos el problema de los registros duplicados.

Para la base que relaciona el deporte con el atleta, debemos hacer lo mismo.

In [112]:
```
data_sports_dp = data_sports.drop_duplicates(subset='Athlete')
```

In [113]:
```
data_final = pd.merge(left = data_main_country_dp, right = data_sports_dp,
                                left_on=key, right_on = key)
```

In [114]:
```
data_final.head()
```

Out[114]:

| | Athlete | Age | Year | Closing Ceremony Date | Gold Medals | Silver Medals | Bronze Medals | Total Medals | Country | Sport |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Michael Phelps | 23.0 | 2008 | 08/24/2008 | 8 | 0 | 0 | 8 | United States | Swimming |
| **1** | Michael Phelps | 19.0 | 2004 | 08/29/2004 | 6 | 0 | 2 | 8 | United States | Swimming |
| **2** | Michael Phelps | 27.0 | 2012 | 08/12/2012 | 4 | 2 | 0 | 6 | United States | Swimming |
| **3** | Natalie Coughlin | 25.0 | 2008 | 08/24/2008 | 1 | 2 | 3 | 6 | United States | Swimming |
| **4** | Natalie Coughlin | 21.0 | 2004 | 08/29/2004 | 2 | 2 | 1 | 5 | United States | Swimming |

Enhorabuena !! Hemos juntados 3 datasets con un campo en común.

# Tipos de Joins (Uniones) teoría

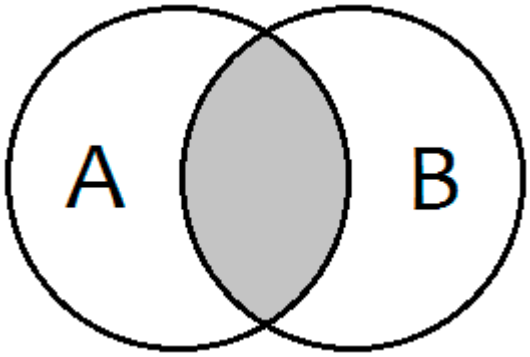
In [115]:
```
from IPython.display import Image
```

## Inner Join

- Devuelve un data frame con las filas que tienen valor tanto en el primero como en el segundo data frame que estamos uniendo
- El número de filas será igual al número de filas comunes que tengan ambos data sets
  - Data Set A tiene 60 gilas
  - Data Set B tiene 50 filas
  - Ambos comparte 30 filas
  - Entonces A Inner Join B tendrá 30 filas
- En términos de teoría de conjuntos, se trata de la intersección de los dos conjuntos

In [116]:
```
Image(filename='/Users/fsanmartin/python-ml-course-master/notebooks/resources/inner-join.png')
```

Out[116]:

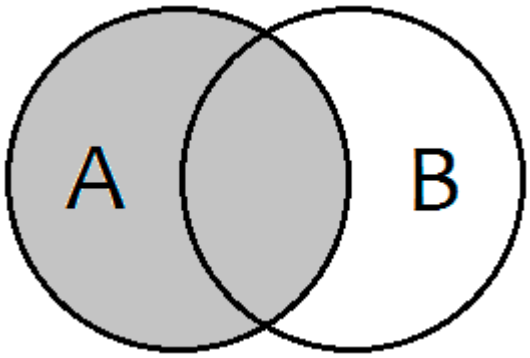



## Left Join

- Devuelve un data frame con las filas que tuvieran valor en el dataset de la izquierda, sin importar si tienen correspondencia en el de la derecha o no
- Las filas del data frame final que no corresponden a ninguna fila del data frame derecho, tendrán NA's en las columnas del data frame derecho
- El número de filas será igual al número de filas del data frame izquierdo
  - Data Set A tiene 60 gilas
  - Data Set B tiene 50 filas
  - Entonces A Left Join B tendrá 60 filas
- En términos de teoría de conjuntos, se trata del propio data set de la izquierda quien, además tiene la intersección en su interior

In [117]:
```
Image(filename='/Users/fsanmartin/python-ml-course-master/notebooks/resources/left-join.png')
```

Out[117]:

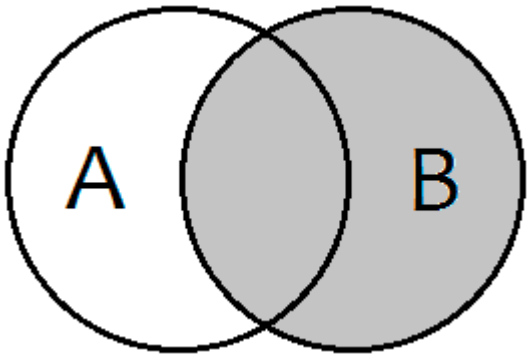


# Right Join

- Devuelve un data frame con las filas que tuvieran valor en el dataset de la derecha, sin importar si tienen correspondencia en el de la izquierda o no
- Las filas del data frame final que no corresponden a ninguna fila del data frame izquierdo, tendrán NA's en las columnas del data frame izquierdo
- El número de filas será igual al número de filas del data frame derecho
  - Data Set A tiene 60 gilas
  - Data Set B tiene 50 filas
  - Entonces A Right Join B tendrá 50 filas
- En términos de teoría de conjuntos, se trata del propio data set de la derecha quien, además tiene la intersección en su interior

In [118]:
```
Image(filename='/Users/fsanmartin/python-ml-course-master/notebooks/resources/right-join.png')
```

Out[118]:

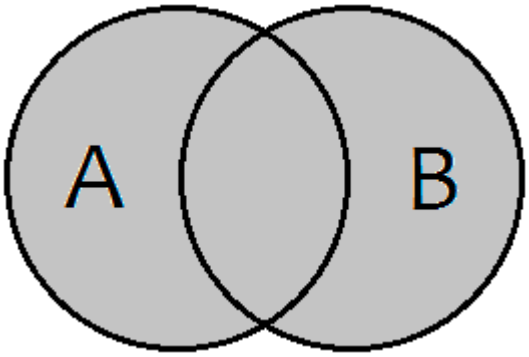



# Outer Join (full)

- Devuelve un data frame con las filas de ambos, reemplazando las ausencias de uno o de otro con NA's en la región específica.
- Las filas del data frame final que no corresponden a ninguna fila del data frame izquierdo (o derecho), tendrán NA's en las columnas del data frame izquierdo (o derecho)
- El número de filas será igual al máximo número de filas de ambos data frames
  - Data Set A tiene 60 gilas
  - Data Set B tiene 50 filas
  - Ambos comparten 30 filas
  - Entonces A Outer Join B tendrá 60 + 50 - 30 = 80 filas
- En términos de teoría de conjuntos, se trata de una unión de conjuntos

In [119]:
```
Image(filename='/Users/fsanmartin/python-ml-course-master/notebooks/resources/outer-join.png')
```

Out[119]:




In [ ]:

In [ ]:

In [ ]:

In [ ]: