09. Pandas Manipulacion Limpieza Visualizacion

August 26, 2024

1 Pandas

1.1 ¿Qué es Pandas?

Pandas es una biblioteca de Python y sirve para analizar datos. Tiene funciones para analizar, limpiar, explorar y manipular datos.

Pandas es una herramienta de análisis y manipulación de datos de código abierto rápida, potente, flexible y fácil de usar, construida sobre el lenguaje de programación Python.

El nombre "Pandas" hace referencia tanto a "Panel Data" como a "Python Data Analysis" y fue creado por Wes McKinney en 2008.

El código fuente de Pandas se encuentra en un repositorio de GitHub.

1.2 Historia de su desarrollo

En 2008, AQR Capital Management comenzó a desarrollar pandas. A fines de 2009, ya era de código abierto y, en la actualidad, cuenta con el apoyo activo de una comunidad de personas con ideas afines en todo el mundo que contribuyen con su valioso tiempo y energía para ayudar a que el código abierto de pandas sea posible. Gracias a todos nuestros colaboradores.

Desde 2015, pandas es un proyecto patrocinado por NumFOCUS. Esto ayudará a garantizar el éxito del desarrollo de pandas como un proyecto de código abierto de clase mundial.

1.3 Documentación

Pandas posee una muy buena documentación en su sitio web oficial. En ella se puede encontrar una guía rápida de uso, además de la documentación completa del paquete.

1.4 ¿Porqué usar Pandas?

Pandas nos permite analizar grandes cantidades de datos y sacar conclusiones basadas en teorías estadísticas.

Pandas puede limpiar conjuntos de datos desordenados y hacerlos legibles y relevantes.

Los datos relevantes son muy importantes en la ciencia de datos.

Pandas puede dar respuestas sobre los datos. Por ejemplo:

- ¿Existe una correlación entre dos o más columnas?
- ¿Qué es el valor promedio?

- ¿Valor máximo?
- ¿Valor mínimo?

Pandas también puede eliminar filas que no son relevantes o que contienen valores incorrectos, como valores vacíos o NULL. Esto se llama limpiar los datos.

1.5 Instalación

Si ya tiene Python y PIP instalados en un sistema, entonces la instalación de Pandas es muy sencilla. Instálelo usando este comando:

```
$ pip install pandas
```

Si este comando falla, entonces use una distribución de Python que ya tenga Pandas instalado, como Anaconda, Spyder, etc.

1.6 Importar Pandas

Una vez que Pandas esté instalado, impórtelo en sus aplicaciones agregando la palabra clave `import`:

```
import pandas
```

Con esto Pandas está importado y listo para usarse.

Ejemplo.

[ ]:
```
import pandas

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}

myvar = pandas.DataFrame(mydataset)

print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

Al importar pandas se puede emplear el alias `pd`. Ahora, el paquete Pandas se puede invocar con `pd` en lugar del nombre completo `pandas`.

[ ]:
```
import pandas as pd

mydataset = {
  'cars': ["BMW", "Volvo", "Ford"],
  'passings': [3, 7, 2]
}
```

```
myvar = pd.DataFrame(mydataset)

print(myvar)
```

```
    cars  passings
0    BMW         3
1  Volvo         7
2   Ford         2
```

1.7 Verificando la versión

La versión de pandas se almacena en una cadena almacenada en el atributo `__version__`.

[ ]:
```
import pandas as pd

print(pd.__version__)
```

```
2.2.2
```

1.8 Series Pandas

Una serie de Pandas es como una columna de una tabla. Es una matriz unidimensional que contiene datos de cualquier tipo.

Ejemplo. Crear una serie de pandas a partir de una lista.

[ ]:
```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a)

print(myvar)
```

```
0    1
1    7
2    2
dtype: int64
```

Si no se especifica nada más, los valores se etiquetan con su número de índice. El primer valor tiene el índice 0, el segundo valor tiene el índice 1, etc.

Esta etiqueta se puede utilizar para acceder a un valor especifico.

Ejemplo. Devolver el primer valor de la serie.

[ ]:
```
print(myvar[0])
```

```
1
```

Con el argumento `index` se puede nombrar las etiquetas.

Ejemplo. Crear etiquetas propias.

[ ]:
```
import pandas as pd

a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

```
x    1
y    7
z    2
dtype: int64
```

Una vez que haya creado etiquetas, podrá acceder a un elemento haciendo referencia a la etiqueta.

Ejemplo. Devolver el valor de "y".

[ ]:
```
print(myvar["y"])
```

```
7
```

1.9 Objetos llave/valor como series

También puedes utilizar un objeto clave/valor, como un diccionario, al crear una Serie.

Ejemplo. Crea una serie Pandas sencilla a partir de un diccionario.

[ ]:
```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories)

print(myvar)
```

```
day1    420
day2    380
day3    390
dtype: int64
```

Observe que las claves del diccionario se convierten en las etiquetas.

Para seleccionar solo algunos de los elementos del diccionario, utilice el argumento de índice y especifique solo los elementos que desea incluir en la serie.

Ejemplo. Cree una serie utilizando solo datos de "día1" y "día2".

[ ]:
```
import pandas as pd

calories = {"day1": 420, "day2": 380, "day3": 390}

myvar = pd.Series(calories, index = ["day1", "day2"])
```

```
print(myvar)
```

```
day1    420
day2    380
dtype: int64
```

1.10 DataFrames

Los conjuntos de datos en Pandas suelen ser tablas multidimensionales, llamadas DataFrames.

Una serie es como una columna, un DataFrame es la tabla completa.

Ejemplo. Crear un DataFrame a partir de dos series.

[ ]:
```
import pandas as pd

# Diccionario data
data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data)

print(df)
```

```
   calories  duration
0       420        50
1       380        40
2       390        45
```

1.11 ¿Qué es un DataFrame?

Un Pandas DataFrame es una estructura de datos de 2 dimensiones, como una de 2 dimensiones array, o una tabla con filas y columnas.

Como puede ver en el resultado anterior, el DataFrame es como una tabla con filas y columnas.

1.12 Localizar una fila

Los pandas usan el atributo `loc` a devolver una o más filas especificadas.

Ejemplo. Devolver la fila 0.

[ ]:
```
print(df.loc[0])
```

```
calories    420
duration     50
Name: 0, dtype: int64
```

Este código devuelve una serie Pandas.

Ejemplo. Devolver filas 0 y 1.

```
[ ]: print(df.loc[[0, 1]])
```

```
   calories  duration
0       420        50
1       380        40
```

En este caso, el código devuelve un DataFrame Pandas.

1.13 Índices nombrados

Con el argumento `index`, se puede nombrar sus propios índices. Nótese que los índices deben ser una lista.

```
[ ]: import pandas as pd

data = {
  "calories": [420, 380, 390],
  "duration": [50, 40, 45]
}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

```
      calories  duration
day1       420        50
day2       380        40
day3       390        45
```

1.14 Localizar Índices Nombrados

Utilice el índice nombrado en el atributo `loc` para devolver la fila especificada.

Ejemplo. Retorno "day2".

```
[ ]: # Referencia al índice nombrado:
print(df.loc["day2"])
```

```
calories    380
duration     40
Name: day2, dtype: int64
```

1.15 Carga de archivos en un DataFrame

Una forma sencilla de almacenar grandes conjuntos de datos es usar archivos CSV (separados por comas archivos).

Los archivos CSV contienen texto sin formato y es un formato bien conocido que puede ser leído por todos, incluyendo Pandas.

Si sus conjuntos de datos se almacenan en un archivo, Pandas puede cargarlos en un DataFrame.

Ejemplo. Cargue un archivo separado por comas (Archivo CSV) en un DataFrame.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

print(df)
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
..        …     …         …       …
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4

[169 rows x 4 columns]
```

Si tiene un DataFrame grande con muchas filas, Pandas solo devolverá las primeras 5 filas y las últimas 5 filas.

Se puede emplear el método `to_string()` para convertir el DataFrame en una sola representación String. La salida es una tabla amigable para la consola.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

print(df.to_string())
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
5          60    102       127     300.0
6          60    110       136     374.0
7          45    104       134     253.3
8          30    109       133     195.1
9          60     98       124     269.0
10         60    103       147     329.3
11         60    100       120     250.7
```

12	60	106	128	345.3
13	60	104	132	379.3
14	60	98	123	275.0
15	60	98	120	215.2
16	60	100	120	300.0
17	45	90	112	NaN
18	60	103	123	323.0
19	45	97	125	243.0
20	60	108	131	364.2
21	45	100	119	282.0
22	60	130	101	300.0
23	45	105	132	246.0
24	60	102	126	334.5
25	60	100	120	250.0
26	60	92	118	241.0
27	60	103	132	NaN
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1
32	60	101	124	299.0
33	60	93	113	223.0
34	60	107	136	361.0
35	60	114	140	415.0
36	60	102	127	300.0
37	60	100	120	300.0
38	60	100	120	300.0
39	45	104	129	266.0
40	45	90	112	180.1
41	60	98	126	286.0
42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0
48	45	108	129	298.0
49	60	111	139	397.6
50	60	107	136	380.2
51	80	123	146	643.1
52	60	106	130	263.0
53	60	118	151	486.0
54	30	136	175	238.0
55	60	121	146	450.7
56	60	118	121	413.0
57	45	115	144	305.0
58	20	153	172	226.4
59	45	123	152	321.0

60	210	108	160	1376.0
61	160	110	137	1034.4
62	160	109	135	853.0
63	45	118	141	341.0
64	20	110	130	131.4
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
68	20	106	136	110.4
69	300	108	143	1500.2
70	150	97	129	1115.0
71	60	109	153	387.6
72	90	100	127	700.0
73	150	97	127	953.2
74	45	114	146	304.0
75	90	98	125	563.2
76	45	105	134	251.0
77	45	110	141	300.0
78	120	100	130	500.4
79	270	100	131	1729.0
80	30	159	182	319.2
81	45	149	169	344.0
82	30	103	139	151.1
83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.0
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3
90	180	101	127	600.1
91	45	107	137	NaN
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4
96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	180	90	120	800.3
107	30	90	120	86.2

108	90	90	120	500.3
109	210	137	184	1860.4
110	60	102	124	325.2
111	45	107	124	275.0
112	15	124	139	124.2
113	45	100	120	225.3
114	60	108	131	367.6
115	60	108	151	351.7
116	60	116	141	443.0
117	60	97	122	277.4
118	60	105	125	NaN
119	60	103	124	332.7
120	30	112	137	193.9
121	45	100	120	100.7
122	60	119	169	336.7
123	60	107	127	344.9
124	60	111	151	368.5
125	60	98	122	271.0
126	60	97	124	275.3
127	60	109	127	382.0
128	90	99	125	466.4
129	60	114	151	384.0
130	60	104	134	342.5
131	60	107	138	357.5
132	60	103	133	335.0
133	60	106	132	327.5
134	60	103	136	339.0
135	20	136	156	189.0
136	45	117	143	317.7
137	45	115	137	318.0
138	45	113	138	308.0
139	20	141	162	222.4
140	60	108	135	390.0
141	60	97	127	NaN
142	45	100	120	250.4
143	45	122	149	335.4
144	60	136	170	470.2
145	45	106	126	270.8
146	60	107	136	400.0
147	60	112	146	361.9
148	30	103	127	185.0
149	60	110	150	409.4
150	60	106	134	343.0
151	60	109	129	353.2
152	60	109	138	374.0
153	30	150	167	275.8
154	60	105	128	328.0
155	60	111	151	368.5

```
156        60     97       131     270.4
157        60    100       120     270.4
158        60    114       150     382.8
159        30     80       120     240.9
160        30     85       120     250.4
161        45     90       130     260.4
162        45     95       130     270.0
163        45    100       140     280.9
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4
```

[ ]:
```
import pandas as pd

df = pd.read_csv('data/social.csv')

print(df.to_string())
```

```
  Social media platform Number of monthly (million)
0              Facebook                      3 000m
1               YouTube                   2000.5 m
2             Instagram                     2 000m
3                TikTok                   1000.5 m
4              Snapchat                      800 m
5           X (Twitter)                      611 m
6                Reddit                      500 m
7             Pinterest                      498 m
8              LinkedIn                      350 m
9               Threads                      175 m
```

1.16 max_filas

El número de filas devueltas se define en la configuración de la opción Pandas. Puede verificar las filas máximas de su sistema con la declaración `pd.options.display.max_rows`.

[ ]:
```
import pandas as pd

print(pd.options.display.max_rows)
```

```
60
```

En mi sistema el número es 60, lo que significa que si el DataFrame contiene más de 60 filas, el `print(df)` la declaración devolverá solo los encabezados y las primeras y últimas 5 filas.

Puede cambiar el número máximo de filas con la misma instrucción.

[ ]:
```
import pandas as pd
```

```
pd.options.display.max_rows = 9999
df = pd.read_csv('data/data.csv')

print(df)
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
5          60    102       127     300.0
6          60    110       136     374.0
7          45    104       134     253.3
8          30    109       133     195.1
9          60     98       124     269.0
10         60    103       147     329.3
11         60    100       120     250.7
12         60    106       128     345.3
13         60    104       132     379.3
14         60     98       123     275.0
15         60     98       120     215.2
16         60    100       120     300.0
17         45     90       112       NaN
18         60    103       123     323.0
19         45     97       125     243.0
20         60    108       131     364.2
21         45    100       119     282.0
22         60    130       101     300.0
23         45    105       132     246.0
24         60    102       126     334.5
25         60    100       120     250.0
26         60     92       118     241.0
27         60    103       132       NaN
28         60    100       132     280.0
29         60    102       129     380.3
30         60     92       115     243.0
31         45     90       112     180.1
32         60    101       124     299.0
33         60     93       113     223.0
34         60    107       136     361.0
35         60    114       140     415.0
36         60    102       127     300.0
37         60    100       120     300.0
38         60    100       120     300.0
39         45    104       129     266.0
40         45     90       112     180.1
41         60     98       126     286.0
```

42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0
48	45	108	129	298.0
49	60	111	139	397.6
50	60	107	136	380.2
51	80	123	146	643.1
52	60	106	130	263.0
53	60	118	151	486.0
54	30	136	175	238.0
55	60	121	146	450.7
56	60	118	121	413.0
57	45	115	144	305.0
58	20	153	172	226.4
59	45	123	152	321.0
60	210	108	160	1376.0
61	160	110	137	1034.4
62	160	109	135	853.0
63	45	118	141	341.0
64	20	110	130	131.4
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
68	20	106	136	110.4
69	300	108	143	1500.2
70	150	97	129	1115.0
71	60	109	153	387.6
72	90	100	127	700.0
73	150	97	127	953.2
74	45	114	146	304.0
75	90	98	125	563.2
76	45	105	134	251.0
77	45	110	141	300.0
78	120	100	130	500.4
79	270	100	131	1729.0
80	30	159	182	319.2
81	45	149	169	344.0
82	30	103	139	151.1
83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.0
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3

90	180	101	127	600.1
91	45	107	137	NaN
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4
96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	180	90	120	800.3
107	30	90	120	86.2
108	90	90	120	500.3
109	210	137	184	1860.4
110	60	102	124	325.2
111	45	107	124	275.0
112	15	124	139	124.2
113	45	100	120	225.3
114	60	108	131	367.6
115	60	108	151	351.7
116	60	116	141	443.0
117	60	97	122	277.4
118	60	105	125	NaN
119	60	103	124	332.7
120	30	112	137	193.9
121	45	100	120	100.7
122	60	119	169	336.7
123	60	107	127	344.9
124	60	111	151	368.5
125	60	98	122	271.0
126	60	97	124	275.3
127	60	109	127	382.0
128	90	99	125	466.4
129	60	114	151	384.0
130	60	104	134	342.5
131	60	107	138	357.5
132	60	103	133	335.0
133	60	106	132	327.5
134	60	103	136	339.0
135	20	136	156	189.0
136	45	117	143	317.7
137	45	115	137	318.0

```
138        45    113        138     308.0
139        20    141        162     222.4
140        60    108        135     390.0
141        60     97        127       NaN
142        45    100        120     250.4
143        45    122        149     335.4
144        60    136        170     470.2
145        45    106        126     270.8
146        60    107        136     400.0
147        60    112        146     361.9
148        30    103        127     185.0
149        60    110        150     409.4
150        60    106        134     343.0
151        60    109        129     353.2
152        60    109        138     374.0
153        30    150        167     275.8
154        60    105        128     328.0
155        60    111        151     368.5
156        60     97        131     270.4
157        60    100        120     270.4
158        60    114        150     382.8
159        30     80        120     240.9
160        30     85        120     250.4
161        45     90        130     260.4
162        45     95        130     270.0
163        45    100        140     280.9
164        60    105        140     290.8
165        60    110        145     300.0
166        60    115        145     310.2
167        75    120        150     320.4
168        75    125        150     330.4
```

1.17 Leer archivos JSON

Los grandes conjuntos de datos a menudo se almacenan o extraen como JSON.

JSON es texto sin formato, pero tiene el formato de un objeto, y es bien conocido en el mundo de la programación, incluyendo Pandas.

En nuestros ejemplos usaremos un archivo JSON llamado `'data.json'`.

Ejemplo. Cargar el archivo `json` en un DataFrame.

[ ]:
```
import pandas as pd

df = pd.read_json('data/data.json')

print(df.to_string())
```

```
     Duration  Pulse  Maxpulse  Calories
```

0	60	110	130	409.1
1	60	117	145	479.0
2	60	103	135	340.0
3	45	109	175	282.4
4	45	117	148	406.0
5	60	102	127	300.5
6	60	110	136	374.0
7	45	104	134	253.3
8	30	109	133	195.1
9	60	98	124	269.0
10	60	103	147	329.3
11	60	100	120	250.7
12	60	106	128	345.3
13	60	104	132	379.3
14	60	98	123	275.0
15	60	98	120	215.2
16	60	100	120	300.0
17	45	90	112	NaN
18	60	103	123	323.0
19	45	97	125	243.0
20	60	108	131	364.2
21	45	100	119	282.0
22	60	130	101	300.0
23	45	105	132	246.0
24	60	102	126	334.5
25	60	100	120	250.0
26	60	92	118	241.0
27	60	103	132	NaN
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1
32	60	101	124	299.0
33	60	93	113	223.0
34	60	107	136	361.0
35	60	114	140	415.0
36	60	102	127	300.5
37	60	100	120	300.1
38	60	100	120	300.0
39	45	104	129	266.0
40	45	90	112	180.1
41	60	98	126	286.0
42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0

48	45	108	129	298.0
49	60	111	139	397.6
50	60	107	136	380.2
51	80	123	146	643.1
52	60	106	130	263.0
53	60	118	151	486.0
54	30	136	175	238.0
55	60	121	146	450.7
56	60	118	121	413.0
57	45	115	144	305.0
58	20	153	172	226.4
59	45	123	152	321.0
60	210	108	160	1376.0
61	160	110	137	1034.4
62	160	109	135	853.0
63	45	118	141	341.0
64	20	110	130	131.4
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
68	20	106	136	110.4
69	300	108	143	1500.2
70	150	97	129	1115.0
71	60	109	153	387.6
72	90	100	127	700.0
73	150	97	127	953.2
74	45	114	146	304.0
75	90	98	125	563.2
76	45	105	134	251.0
77	45	110	141	300.0
78	120	100	130	500.4
79	270	100	131	1729.0
80	30	159	182	319.2
81	45	149	169	344.0
82	30	103	139	151.1
83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.1
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3
90	180	101	127	600.1
91	45	107	137	NaN
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4

96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	180	90	120	800.3
107	30	90	120	86.2
108	90	90	120	500.3
109	210	137	184	1860.4
110	60	102	124	325.2
111	45	107	124	275.0
112	15	124	139	124.2
113	45	100	120	225.3
114	60	108	131	367.6
115	60	108	151	351.7
116	60	116	141	443.0
117	60	97	122	277.4
118	60	105	125	NaN
119	60	103	124	332.7
120	30	112	137	193.9
121	45	100	120	100.7
122	60	119	169	336.7
123	60	107	127	344.9
124	60	111	151	368.5
125	60	98	122	271.0
126	60	97	124	275.3
127	60	109	127	382.0
128	90	99	125	466.4
129	60	114	151	384.0
130	60	104	134	342.5
131	60	107	138	357.5
132	60	103	133	335.0
133	60	106	132	327.5
134	60	103	136	339.0
135	20	136	156	189.0
136	45	117	143	317.7
137	45	115	137	318.0
138	45	113	138	308.0
139	20	141	162	222.4
140	60	108	135	390.0
141	60	97	127	NaN
142	45	100	120	250.4
143	45	122	149	335.4

```
144         60    136         170      470.2
145         45    106         126      270.8
146         60    107         136      400.0
147         60    112         146      361.9
148         30    103         127      185.0
149         60    110         150      409.4
150         60    106         134      343.0
151         60    109         129      353.2
152         60    109         138      374.0
153         30    150         167      275.8
154         60    105         128      328.0
155         60    111         151      368.5
156         60     97         131      270.4
157         60    100         120      270.4
158         60    114         150      382.8
159         30     80         120      240.9
160         30     85         120      250.4
161         45     90         130      260.4
162         45     95         130      270.0
163         45    100         140      280.9
164         60    105         140      290.8
165         60    110         145      300.4
166         60    115         145      310.2
167         75    120         150      320.4
168         75    125         150      330.4
```

Los objetos JSON tienen el mismo formato que los diccionarios Python.

Si su código JSON no está en un archivo, sino en un diccionario Python, puede cargarlo en un DataFrame directamente.

Ejemplo. Cargar un diccionario Python en un DataFrame.

[ ]:
```python
import pandas as pd

data = {
  "Duration":{
    "0":60,
    "1":60,
    "2":60,
    "3":45,
    "4":45,
    "5":60
  },
  "Pulse":{
    "0":110,
    "1":117,
    "2":103,
    "3":109,
```

```
    "4":117,
    "5":102
  },
  "Maxpulse":{
    "0":130,
    "1":145,
    "2":135,
    "3":175,
    "4":148,
    "5":127
  },
  "Calories":{
    "0":409,
    "1":479,
    "2":340,
    "3":282,
    "4":406,
    "5":300
  }
}

df = pd.DataFrame(data)

print(df)
```

```
   Duration  Pulse  Maxpulse  Calories
0        60    110       130       409
1        60    117       145       479
2        60    103       135       340
3        45    109       175       282
4        45    117       148       406
5        60    102       127       300
```

1.18 Vistazo rápido a los Datos

Uno de los métodos más utilizados para obtener una visión general rápida del DataFrame, es el `head()` método.

El `head()` el método devuelve los encabezados y un número específico de filas, comenzando desde la parte superior. Por defecto se muestran las primeras 5 líneas a menos que se indique otra cosa.

Ejemplo. Obtenga una visión general rápida imprimiendo las primeras 10 filas de DataFrame.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

print(df.head(10))
```

```
   Duration  Pulse  Maxpulse  Calories
0        60    110       130     409.1
1        60    117       145     479.0
2        60    103       135     340.0
3        45    109       175     282.4
4        45    117       148     406.0
5        60    102       127     300.0
6        60    110       136     374.0
7        45    104       134     253.3
8        30    109       133     195.1
9        60     98       124     269.0
```

También hay un `tail()` método para ver el último filas del DataFrame.

El `tail()` el método devuelve los encabezados y un número específico de filas, comenzando desde la parte inferior.

Ejemplo. Imprima las últimas 5 filas de DataFrame.

[ ]:
```
print(df.tail())
```

```
     Duration  Pulse  Maxpulse  Calories
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4
```

1.19 Información Sobre los Datos

El objeto DataFrames tiene un método llamado `info()`, eso le da más información sobre el conjunto de datos.

Ejemplo. Imprimir información sobre los datos.

[ ]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  169 non-null    int64
 1   Pulse     169 non-null    int64
 2   Maxpulse  169 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
```

El resultado nos dice que hay 169 filas y 4 columnas, el nombre de cada columna, con el tipo de datos.

El `info()` el método también nos dice cuántos valores no nulos hay presentes en cada columna y en nuestro conjunto de datos parece que hay 164 de 169 valores No Nulos en la columna "Calorías".

Lo que significa que hay 5 filas sin ningún valor en absoluto, en la columna "Calorías", por alguna razón desconocida.

Los valores vacíos, o valores nulos, pueden ser malos al analizar los datos y debe considerar eliminar filas con valores vacíos. Este es un paso hacia lo que se llama datos de limpieza.

2 Limpieza de Datos

La limpieza de datos significa arreglar datos incorrectos en su conjunto de datos.

Los datos malos incorrectos pueden ser:

- Celdas vacías
- Datos en formato incorrecto
- Datos incorrectos
- Duplicados

Las celdas vacías pueden potencialmente darle un resultado incorrecto cuando analiza datos.

2.1 Eliminar Filas

Una forma de tratar con las celdas vacías es eliminar las filas que las contienen.

Esto suele estar bien, siempre y cuando los conjuntos de datos sean muy grandes, ya que eliminar algunas filas no tendrá un gran impacto en el resultado.

Ejemplo. Devuelve un nuevo Marco de datos sin celdas vacías.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

new_df = df.dropna()

print(new_df.to_string())
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
5          60    102       127     300.0
6          60    110       136     374.0
7          45    104       134     253.3
8          30    109       133     195.1
9          60     98       124     269.0
10         60    103       147     329.3
11         60    100       120     250.7
```

12	60	106	128	345.3
13	60	104	132	379.3
14	60	98	123	275.0
15	60	98	120	215.2
16	60	100	120	300.0
18	60	103	123	323.0
19	45	97	125	243.0
20	60	108	131	364.2
21	45	100	119	282.0
22	60	130	101	300.0
23	45	105	132	246.0
24	60	102	126	334.5
25	60	100	120	250.0
26	60	92	118	241.0
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1
32	60	101	124	299.0
33	60	93	113	223.0
34	60	107	136	361.0
35	60	114	140	415.0
36	60	102	127	300.0
37	60	100	120	300.0
38	60	100	120	300.0
39	45	104	129	266.0
40	45	90	112	180.1
41	60	98	126	286.0
42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0
48	45	108	129	298.0
49	60	111	139	397.6
50	60	107	136	380.2
51	80	123	146	643.1
52	60	106	130	263.0
53	60	118	151	486.0
54	30	136	175	238.0
55	60	121	146	450.7
56	60	118	121	413.0
57	45	115	144	305.0
58	20	153	172	226.4
59	45	123	152	321.0
60	210	108	160	1376.0
61	160	110	137	1034.4

62	160	109	135	853.0
63	45	118	141	341.0
64	20	110	130	131.4
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
68	20	106	136	110.4
69	300	108	143	1500.2
70	150	97	129	1115.0
71	60	109	153	387.6
72	90	100	127	700.0
73	150	97	127	953.2
74	45	114	146	304.0
75	90	98	125	563.2
76	45	105	134	251.0
77	45	110	141	300.0
78	120	100	130	500.4
79	270	100	131	1729.0
80	30	159	182	319.2
81	45	149	169	344.0
82	30	103	139	151.1
83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.0
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3
90	180	101	127	600.1
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4
96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	180	90	120	800.3
107	30	90	120	86.2
108	90	90	120	500.3
109	210	137	184	1860.4
110	60	102	124	325.2

111	45	107	124	275.0
112	15	124	139	124.2
113	45	100	120	225.3
114	60	108	131	367.6
115	60	108	151	351.7
116	60	116	141	443.0
117	60	97	122	277.4
119	60	103	124	332.7
120	30	112	137	193.9
121	45	100	120	100.7
122	60	119	169	336.7
123	60	107	127	344.9
124	60	111	151	368.5
125	60	98	122	271.0
126	60	97	124	275.3
127	60	109	127	382.0
128	90	99	125	466.4
129	60	114	151	384.0
130	60	104	134	342.5
131	60	107	138	357.5
132	60	103	133	335.0
133	60	106	132	327.5
134	60	103	136	339.0
135	20	136	156	189.0
136	45	117	143	317.7
137	45	115	137	318.0
138	45	113	138	308.0
139	20	141	162	222.4
140	60	108	135	390.0
142	45	100	120	250.4
143	45	122	149	335.4
144	60	136	170	470.2
145	45	106	126	270.8
146	60	107	136	400.0
147	60	112	146	361.9
148	30	103	127	185.0
149	60	110	150	409.4
150	60	106	134	343.0
151	60	109	129	353.2
152	60	109	138	374.0
153	30	150	167	275.8
154	60	105	128	328.0
155	60	111	151	368.5
156	60	97	131	270.4
157	60	100	120	270.4
158	60	114	150	382.8
159	30	80	120	240.9
160	30	85	120	250.4

```
161        45     90       130     260.4
162        45     95       130     270.0
163        45    100       140     280.9
164        60    105       140     290.8
165        60    110       145     300.0
166        60    115       145     310.2
167        75    120       150     320.4
168        75    125       150     330.4
```

[ ]:
```
print(new_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 164 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  164 non-null    int64
 1   Pulse     164 non-null    int64
 2   Maxpulse  164 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 6.4 KB
None
```

Por defecto, el método `dropna()` devuelve un nuevo DataFrame, y no cambia el original.

Si desea cambiar el DataFrame original, utilice el argumento `inplace = True`.

Ejemplo. Eliminar todas las filas con valores NULL.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

df.dropna(inplace = True)

print(df.to_string())
```

```
     Duration  Pulse  Maxpulse  Calories
0          60    110       130     409.1
1          60    117       145     479.0
2          60    103       135     340.0
3          45    109       175     282.4
4          45    117       148     406.0
5          60    102       127     300.0
6          60    110       136     374.0
7          45    104       134     253.3
8          30    109       133     195.1
9          60     98       124     269.0
10         60    103       147     329.3
```

11	60	100	120	250.7
12	60	106	128	345.3
13	60	104	132	379.3
14	60	98	123	275.0
15	60	98	120	215.2
16	60	100	120	300.0
18	60	103	123	323.0
19	45	97	125	243.0
20	60	108	131	364.2
21	45	100	119	282.0
22	60	130	101	300.0
23	45	105	132	246.0
24	60	102	126	334.5
25	60	100	120	250.0
26	60	92	118	241.0
28	60	100	132	280.0
29	60	102	129	380.3
30	60	92	115	243.0
31	45	90	112	180.1
32	60	101	124	299.0
33	60	93	113	223.0
34	60	107	136	361.0
35	60	114	140	415.0
36	60	102	127	300.0
37	60	100	120	300.0
38	60	100	120	300.0
39	45	104	129	266.0
40	45	90	112	180.1
41	60	98	126	286.0
42	60	100	122	329.4
43	60	111	138	400.0
44	60	111	131	397.0
45	60	99	119	273.0
46	60	109	153	387.6
47	45	111	136	300.0
48	45	108	129	298.0
49	60	111	139	397.6
50	60	107	136	380.2
51	80	123	146	643.1
52	60	106	130	263.0
53	60	118	151	486.0
54	30	136	175	238.0
55	60	121	146	450.7
56	60	118	121	413.0
57	45	115	144	305.0
58	20	153	172	226.4
59	45	123	152	321.0
60	210	108	160	1376.0

61	160	110	137	1034.4
62	160	109	135	853.0
63	45	118	141	341.0
64	20	110	130	131.4
65	180	90	130	800.4
66	150	105	135	873.4
67	150	107	130	816.0
68	20	106	136	110.4
69	300	108	143	1500.2
70	150	97	129	1115.0
71	60	109	153	387.6
72	90	100	127	700.0
73	150	97	127	953.2
74	45	114	146	304.0
75	90	98	125	563.2
76	45	105	134	251.0
77	45	110	141	300.0
78	120	100	130	500.4
79	270	100	131	1729.0
80	30	159	182	319.2
81	45	149	169	344.0
82	30	103	139	151.1
83	120	100	130	500.0
84	45	100	120	225.3
85	30	151	170	300.0
86	45	102	136	234.0
87	120	100	157	1000.1
88	45	129	103	242.0
89	20	83	107	50.3
90	180	101	127	600.1
92	30	90	107	105.3
93	15	80	100	50.5
94	20	150	171	127.4
95	20	151	168	229.4
96	30	95	128	128.2
97	25	152	168	244.2
98	30	109	131	188.2
99	90	93	124	604.1
100	20	95	112	77.7
101	90	90	110	500.0
102	90	90	100	500.0
103	90	90	100	500.4
104	30	92	108	92.7
105	30	93	128	124.0
106	180	90	120	800.3
107	30	90	120	86.2
108	90	90	120	500.3
109	210	137	184	1860.4

110	60	102	124	325.2
111	45	107	124	275.0
112	15	124	139	124.2
113	45	100	120	225.3
114	60	108	131	367.6
115	60	108	151	351.7
116	60	116	141	443.0
117	60	97	122	277.4
119	60	103	124	332.7
120	30	112	137	193.9
121	45	100	120	100.7
122	60	119	169	336.7
123	60	107	127	344.9
124	60	111	151	368.5
125	60	98	122	271.0
126	60	97	124	275.3
127	60	109	127	382.0
128	90	99	125	466.4
129	60	114	151	384.0
130	60	104	134	342.5
131	60	107	138	357.5
132	60	103	133	335.0
133	60	106	132	327.5
134	60	103	136	339.0
135	20	136	156	189.0
136	45	117	143	317.7
137	45	115	137	318.0
138	45	113	138	308.0
139	20	141	162	222.4
140	60	108	135	390.0
142	45	100	120	250.4
143	45	122	149	335.4
144	60	136	170	470.2
145	45	106	126	270.8
146	60	107	136	400.0
147	60	112	146	361.9
148	30	103	127	185.0
149	60	110	150	409.4
150	60	106	134	343.0
151	60	109	129	353.2
152	60	109	138	374.0
153	30	150	167	275.8
154	60	105	128	328.0
155	60	111	151	368.5
156	60	97	131	270.4
157	60	100	120	270.4
158	60	114	150	382.8
159	30	80	120	240.9

```
160         30      85        120      250.4
161         45      90        130      260.4
162         45      95        130      270.0
163         45     100        140      280.9
164         60     105        140      290.8
165         60     110        145      300.0
166         60     115        145      310.2
167         75     120        150      320.4
168         75     125        150      330.4
```

Ahora, el `dropna(inplace = True)` NO devolverá un DataFrame nuevo, pero eliminará todas las filas que contengan valores `NULL` del DataFrame original.

2.2 Reemplazar Valores Vacíos

Otra forma de tratar con celdas vacías es insertar un nuevo valor en su lugar.

De esta manera, no tiene que eliminar filas enteras solo por algunas celdas vacías.

El `fillna()` el método nos permite reemplazar vacío celdas con un valor.

Ejemplo. Reemplace los valores NULL con el número 130.

[ ]:
```python
import pandas as pd

df = pd.read_csv('data/data.csv')

df.fillna(130, inplace = True)

print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Duration  169 non-null    int64  
 1   Pulse     169 non-null    int64  
 2   Maxpulse  169 non-null    int64  
 3   Calories  169 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
```

2.3 Reemplazar Solo Para Columnas Especificadas

El ejemplo anterior reemplaza todas las celdas vacías en todo el conjutno de datos.

Para reemplazar solo valores vacíos para una columna, se puede especificar el nombre de la columna para el DataFrame.

Ejemplo. Reemplace los valores NULL en las columnas “Calorías” con el número 130.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

df["Calories"].fillna(130, inplace = True)
```

```
/tmp/ipykernel_502828/10114066.py:5: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.


  df["Calories"].fillna(130, inplace = True)
```

2.4 Reemplazar usando Media, Mediana o Moda

Una forma común de reemplazar las celdas vacías es calcular la media, la mediana o la moda de la columna.

Pandas usa los métodos `mean()`, `median()` y `mode()` métodos para calcular los valores respectivos para una columna especificada.

Ejemplo. Calcule la media y reemplazar cualquier valor vacío con ella.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

x = df["Calories"].mean()

df["Calories"].fillna(x, inplace = True)
```

```
/tmp/ipykernel_502828/3993554943.py:7: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an
inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.
```

```
df["Calories"].fillna(x, inplace = True)
```

Ejemplo. Calcular la mediana y reemplazar cualquier valor vacío con ella.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

x = df["Calories"].median()

df["Calories"].fillna(x, inplace = True)
```

```
/tmp/ipykernel_502828/1269507780.py:7: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an
inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.
```

```
df["Calories"].fillna(x, inplace = True)
```

Ejemplo. Calcule la moda y reemplazar cualquier valor vacío con ella.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

x = df["Calories"].mode()[0]

df["Calories"].fillna(x, inplace = True)
```

```
/tmp/ipykernel_502828/2697854522.py:7: FutureWarning: A value is trying to be
set on a copy of a DataFrame or Series through chained assignment using an
inplace method.
The behavior will change in pandas 3.0. This inplace method will never work
because the intermediate object on which we are setting values always behaves as
a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using
'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value)
instead, to perform the operation inplace on the original object.
```

```
df["Calories"].fillna(x, inplace = True)
```

2.5 Datos de Formato Incorrecto

Las celdas de datos con formato incorrecto pueden dificultar, o incluso imposibilitar, el análisis de datos.

Para solucionarlo, tiene dos opciones: eliminar las filas o convertir todas las celdas en el columnas en el mismo formato.

2.6 Convertir en un Formato Correcto

En nuestro conjunto de datos, tenemos dos celdas con el formato incorrecto. Echa un vistazo a la fila 22 y 26, la columna 'Fecha' debe ser una cadena que represente una fecha.

```
Duration Date Pulse Maxpulse Calories
  0 60 '2020/12/01' 110 130 409,1
  1 60 '2020/12/02' 117 145 479,0
  2 60 '2020/12/03' 103 135 340,0
  3 45 '2020/12/04' 109 175 282,4
  4 45 '2020/12/05' 117 148 406,0
  5 60 '2020/12/06' 102 127 300,0
  6 60 '2020/12/07' 110 136 374,0
  7 450 '2020/12/08' 104 134 253,3
  8 30 '2020/12/09' 109 133 195,1
  9 60 '2020/12/10' 98 124 269,0
  10 60 '2020/12/11' 103 147 329,3
  11 60 '2020/12/12' 100 120 250,7
  12 60 '2020/12/12' 100 120 250,7
  13 60 '2020/12/13' 106 128 345,3
  14 60 '2020/12/14' 104 132 379,3
  15 60 '2020/12/15' 98 123 275,0
  16 60 '2020/12/16' 98 120 215,2
  17 60 '2020/12/17' 100 120 300,0
  18 45 '2020/12/18' 90 112 NaN
  19 60 '2020/12/19' 103 123 323,0
  20 45 '2020/12/20' 97 125 243,0
  21 60 '2020/12/21' 108 131 364,2
  22 45 NaN 100 119 282,0
  23 60 '2020/12/23' 130 101 300,0
  24 45 '2020/12/24' 105 132 246,0
  25 60 '2020/12/25' 102 126 334,5
  26 60 20201226 100 120 250,0
  27 60 '2020/12/27' 92 118 241,0
  28 60 '2020/12/28' 103 132 NaN
  29 60 '2020/12/29' 100 132 280,0
  30 60 '2020/12/30' 102 129 380,3
  31 60 '2020/12/31' 92 115 243,0
```

Intentemos convertir todas las celdas de la columna 'Fecha' en fechas. Pandas tiene un método `to_datetime()` para esto.

Ejemplo. Convertir fecha.

```
import pandas as pd

df = pd.read_csv('data.csv')

df['Date'] = pd.to_datetime(df['Date'])

print(df.to_string())
```

Resultado:

```
Duración Fecha Pulso Maxpulse Calorías
  0 60 '2020/12/01' 110 130 409,1
  1 60 '2020/12/02' 117 145 479,0
  2 60 '2020/12/03' 103 135 340,0
  3 45 '2020/12/04' 109 175 282,4
  4 45 '2020/12/05' 117 148 406,0
  5 60 '2020/12/06' 102 127 300,0
  6 60 '2020/12/07' 110 136 374,0
  7 450 '2020/12/08' 104 134 253,3
  8 30 '2020/12/09' 109 133 195,1
  9 60 '2020/12/10' 98 124 269,0
  10 60 '2020/12/11' 103 147 329,3
  11 60 '2020/12/12' 100 120 250,7
  12 60 '2020/12/12' 100 120 250,7
  13 60 '2020/12/13' 106 128 345,3
  14 60 '2020/12/14' 104 132 379,3
  15 60 '2020/12/15' 98 123 275,0
  16 60 '2020/12/16' 98 120 215,2
  17 60 '2020/12/17' 100 120 300,0
  18 45 '2020/12/18' 90 112 NaN
  19 60 '2020/12/19' 103 123 323,0
  20 45 '2020/12/20' 97 125 243,0
  21 60 '2020/12/21' 108 131 364,2
  22 45 NaT 100 119 282,0
  23 60 '2020/12/23' 130 101 300,0
  24 45 '2020/12/24' 105 132 246,0
  25 60 '2020/12/25' 102 126 334,5
  26 60 '2020/12/26' 100 120 250,0
  27 60 '2020/12/27' 92 118 241,0
  28 60 '2020/12/28' 103 132 NaN
  29 60 '2020/12/29' 100 132 280,0
  30 60 '2020/12/30' 102 129 380,3
  31 60 '2020/12/31' 92 115 243,0
```

```
[ ]: import pandas as pd

df = pd.read_csv('data/calorias.csv')

print(df)
```

```
    Unnamed: 0  Duration          Date  Pulse  Maxpulse Calories
0            0        60  '2020/12/01'    110       130    409,1
1            1        60  '2020/12/02'    117       145    479,0
2            2        60  '2020/12/03'    103       135    340,0
3            3        45  '2020/12/04'    109       175    282,4
4            4        45  '2020/12/05'    117       148    406,0
5            5        60  '2020/12/06'    102       127    300,0
6            6        60  '2020/12/07'    110       136    374,0
7            7       450  '2020/12/08'    104       134    253,3
8            8        30  '2020/12/09'    109       133    195,1
9            9        60  '2020/12/10'     98       124    269,0
10          10        60  '2020/12/11'    103       147    329,3
11          11        60  '2020/12/12'    100       120    250,7
12          12        60  '2020/12/12'    100       120    250,7
13          13        60  '2020/12/13'    106       128    345,3
14          14        60  '2020/12/14'    104       132    379,3
15          15        60  '2020/12/15'     98       123    275,0
16          16        60  '2020/12/16'     98       120    215,2
17          17        60  '2020/12/17'    100       120    300,0
18          18        45  '2020/12/18'     90       112      NaN
19          19        60  '2020/12/19'    103       123    323,0
20          20        45  '2020/12/20'     97       125    243,0
21          21        60  '2020/12/21'    108       131    364,2
22          22        45           NaN    100       119    282,0
23          23        60  '2020/12/23'    130       101    300,0
24          24        45  '2020/12/24'    105       132    246,0
25          25        60  '2020/12/25'    102       126    334,5
26          26        60      20201226    100       120    250,0
27          27        60  '2020/12/27'     92       118    241,0
28          28        60  '2020/12/28'    103       132      NaN
29          29        60  '2020/12/29'    100       132    280,0
30          30        60  '2020/12/30'    102       129    380,3
31          31        60  '2020/12/31'     92       115    243,0
```

Como puede ver en el resultado, la fecha en la fila 26 fue fija, pero la fecha vacía en la fila 22 obtuvo un valor de `NaT` (No un tiempo), en otras palabras, un valor vacío. Una forma de lidiar con los valores vacíos es simplemente eliminar toda la fila.

2.7 Borrar Filas

El resultado de la conversión en el ejemplo anterior nos dio un valor `NaT`, que se puede manejar como un valor `NULL`, y podemos eliminar la fila utilizando el método `dropna()`.

Ejemplo. Eliminar filas con un valor NULL en la columna “Fecha”.

```
df.dropna(subset=['Date'], inplace = True)
```

2.8 Datos Incorrectos

Los “Datos incorrectos” no tienen que ser “celdas vacías” o “formato incorrecto”, puede ser solo equívoco, como si alguien registrara “199” en lugar de “1.99”.

A veces puede detectar datos incorrectos mirando el conjunto de datos, porque tiene una expectativa de qué debería ser.

Si echa un vistazo a nuestro conjunto de datos, puede ver que en la fila 7, la duración es 450, pero para todas las demás filas la duración es entre 30 y 60.

No tiene que estar mal, pero teniendo en cuenta que este es el conjunto de datos del entrenamiento de alguien sesiones, concluimos con el hecho de que esta persona no funcionó en 450 minutos.

	Duration	Date	Pulse	Maxpulse	Calories
0	60	'2020/12/01'	110	130	409.1
1	60	'2020/12/02'	117	145	479.0
2	60	'2020/12/03'	103	135	340.0
3	45	'2020/12/04'	109	175	282.4
4	45	'2020/12/05'	117	148	406.0
5	60	'2020/12/06'	102	127	300.0
6	60	'2020/12/07'	110	136	374.0
7	450	'2020/12/08'	104	134	253.3
8	30	'2020/12/09'	109	133	195.1
9	60	'2020/12/10'	98	124	269.0
10	60	'2020/12/11'	103	147	329.3
11	60	'2020/12/12'	100	120	250.7
12	60	'2020/12/12'	100	120	250.7
13	60	'2020/12/13'	106	128	345.3
14	60	'2020/12/14'	104	132	379.3
15	60	'2020/12/15'	98	123	275.0
16	60	'2020/12/16'	98	120	215.2
17	60	'2020/12/17'	100	120	300.0
18	45	'2020/12/18'	90	112	NaN
19	60	'2020/12/19'	103	123	323.0
20	45	'2020/12/20'	97	125	243.0
21	60	'2020/12/21'	108	131	364.2
22	45	NaN	100	119	282.0
23	60	'2020/12/23'	130	101	300.0
24	45	'2020/12/24'	105	132	246.0
25	60	'2020/12/25'	102	126	334.5
26	60	20201226	100	120	250.0
27	60	'2020/12/27'	92	118	241.0
28	60	'2020/12/28'	103	132	NaN
29	60	'2020/12/29'	100	132	280.0
30	60	'2020/12/30'	102	129	380.3

```
  31        60   '2020/12/31'     92        115      243.0
```

¿Cómo podemos corregir valores incorrectos, como el de "Duración" en la fila 7?

2.9 Sustitución de Valores

Una forma de corregir valores incorrectos es reemplazarlos con otra cosa.

En nuestro ejemplo, lo más probable es un error tipográfico, y el valor debe ser "45" en lugar de "450", y nosotros podría insertar "45" en la fila 7:

Ejemplo. Establecer "Duración" = 45 en la fila 7.

```
df.loc[7, 'Duration'] = 45
```

Para conjuntos de datos pequeños, es posible que pueda reemplazar los datos incorrectos uno por uno pero no para grandes conjuntos de datos.

Para reemplazar datos incorrectos para conjuntos de datos más grandes, puede crear algunas reglas, p. establezca algunos límites para los valores legales y reemplace cualquier valor que esté fuera del límites.

Ejemplo. Recorre todos los valores en la columna "Duración", si el valor es superior a 120, establezca en 120.

```
for x in df.index:
    if df.loc[x, "Duration"] > 120:
        df.loc[x, "Duration"] = 120
```

2.10 Eliminación de Filas

Otra forma de manejar datos incorrectos es eliminar las filas que contienen datos incorrectos.

De esta manera no tiene que averiguar con qué reemplazarlos, y los hay una buena oportunidad de que no los necesite para hacer sus análisis.

Ejemplo. Eliminar filas donde "Duración" es superior a 120.

```
for x in df.index:
    if df.loc[x, "Duration"] > 120:
        df.drop(x, inplace = True)
```

2.11 Datos Duplicados

Las filas duplicadas son filas que se han registrado más de una vez.

```
Duración Fecha Pulso Maxpulse Calorías
0 60 '2020/12/01' 110 130 409,1
1 60 '2020/12/02' 117 145 479,0
2 60 '2020/12/03' 103 135 340,0
3 45 '2020/12/04' 109 175 282,4
4 45 '2020/12/05' 117 148 406,0
5 60 '2020/12/06' 102 127 300,0
6 60 '2020/12/07' 110 136 374,0
7 450 '2020/12/08' 104 134 253,3
```

```
8 30 '2020/12/09' 109 133 195,1
9 60 '2020/12/10' 98 124 269,0
10 60 '2020/12/11' 103 147 329,3
11 60 '2020/12/12' 100 120 250,7
12 60 '2020/12/12' 100 120 250,7
13 60 '2020/12/13' 106 128 345,3
14 60 '2020/12/14' 104 132 379,3
15 60 '2020/12/15' 98 123 275,0
16 60 '2020/12/16' 98 120 215,2
17 60 '2020/12/17' 100 120 300,0
18 45 '2020/12/18' 90 112 NaN
19 60 '2020/12/19' 103 123 323,0
20 45 '2020/12/20' 97 125 243,0
21 60 '2020/12/21' 108 131 364,2
22 45 NaN 100 119 282,0
23 60 '2020/12/23' 130 101 300,0
24 45 '2020/12/24' 105 132 246,0
25 60 '2020/12/25' 102 126 334,5
26 60 20201226 100 120 250,0
27 60 '2020/12/27' 92 118 241,0
28 60 '2020/12/28' 103 132 NaN
29 60 '2020/12/29' 100 132 280,0
30 60 '2020/12/30' 102 129 380,3
31 60 '2020/12/31' 92 115 243,0
```

Al echar un vistazo a nuestro conjunto de datos de prueba, podemos suponer que las **filas 11 y 12 son duplicados**.

Para descubrir duplicados, podemos usar el método `duplicated()`.

El método `duplicated()` devuelve valores booleanos para cada fila.

Ejemplo. Devuelve `True` por cada fila que es un duplicado, de otra manera False.

```
print(df.duplicated())
```

3 Correlación de los datos

Un gran aspecto del módulo Pandas es el método `corr()`. Este método calcula la relación entre cada columna en su conjunto de datos.

Ejemplo. Mostrar la relación entre las columnas.

[ ]:
```
import pandas as pd

df = pd.read_csv('data/data.csv')

print(df.corr())
```

```
          Duration     Pulse  Maxpulse  Calories
Duration  1.000000 -0.155408  0.009403  0.922717
```

```
Pulse     -0.155408  1.000000  0.786535  0.025121
Maxpulse   0.009403  0.786535  1.000000  0.203813
Calories   0.922717  0.025121  0.203813  1.000000
```

El resultado del método `corr()` es una tabla con muchos números que representa qué tan bien está la relación entre dos columnas.

El número varía de - 1 a 1. Si la correlación vale 1 significa que hay una relación de 1 a 1 (una correlación perfecta) y para este conjunto de datos, cada vez que un valor sube en la primer columna, el otro valor también subía.

Un valor 0.9 también es una buena relación, y si aumenta un valor, el otro probablemente también aumentará.

Para un valor -0.9 habría tan buena relación como con 0.9, pero en este casi si aumenta un valor, el otro probablemente bajará.

Por último, 0.2 no es una buena relación, lo que significa que si un valor sube no significa que el otro lo haga.

¿Qué es una buena correlación? Depende del uso, pero creo que es seguro decir que tienes que tener al menos 0.6 (o -0.6) para llamarlo una buena correlación.

3.0.1 Correlación Perfecta:

Podemos ver que "Duración" y "Duración" obtuvieron el número 1.000000, lo que tiene sentido cada columna siempre tiene una relación perfecta consigo misma.

3.0.2 Buena Correlación:

"Duración" y "Calorías" tienen un 0.922721 correlación, lo cual es una muy buena correlación, y podemos predecir que cuanto más tiempo trabaje fuera, cuantas más calorías queme, y al revés: si quemó mucho de calorías, probablemente tuviste un largo ejercicio.

3.0.3 Mala Correlación:

"Duración" y "Maxpulse" tiene un 0.009403 correlación, lo cual es una correlación muy mala, lo que significa que no podemos predecir el pulso máximo con solo mirar la duración del ejercicio, y viceversa.

4 Visualización de datos

Pandas utiliza el métpdp `plot()` para crear gráficas.

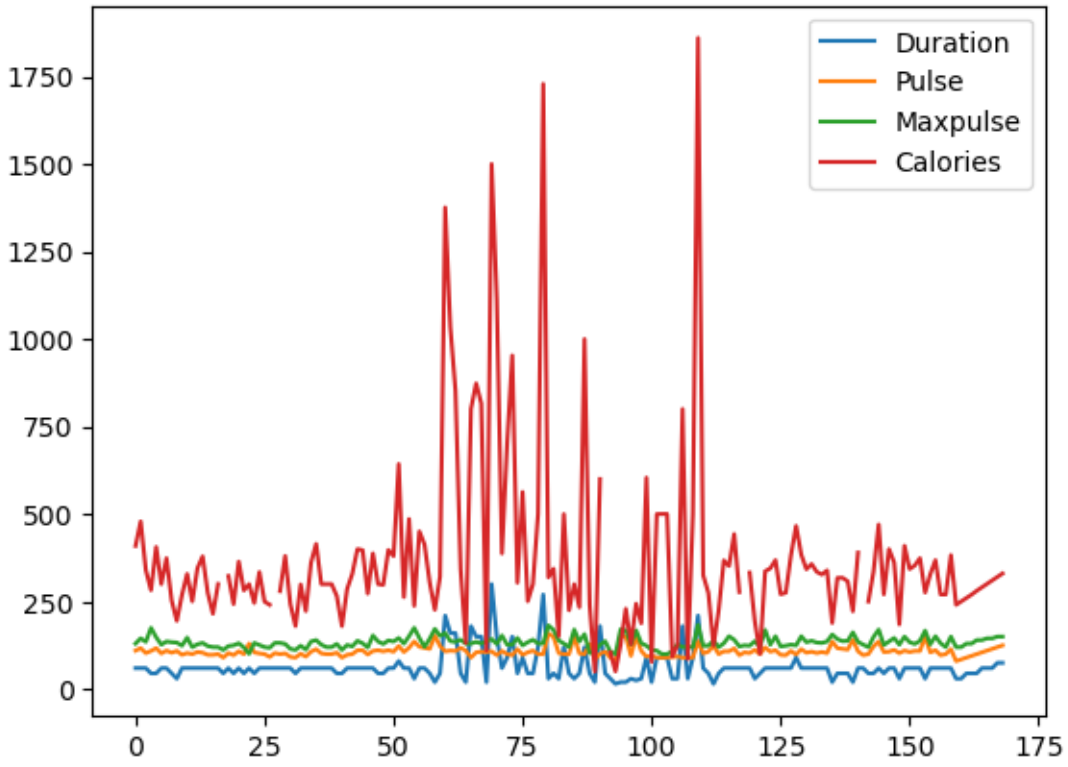
Podemos utilizar `Pyplot`, un submódulo de la biblioteca `Matplotlib` para visualizar el diagrama en la pantalla.

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data/data.csv')
```

```
df.plot()

plt.show()
```



4.1 Diagrama de dispersión

Especifique que desea un gráfico de dispersión con el argumento `kind`.

`kind = 'scatter'`

Un diagrama de dispersión necesita un eje x y un eje y.

En el siguiente ejemplo, utilizaremos "Duración" para el eje x y "Calorías" para el eje y.

Incluya los argumentos x e y de la siguiente manera:

`x = 'Duration', y = 'Calories'`

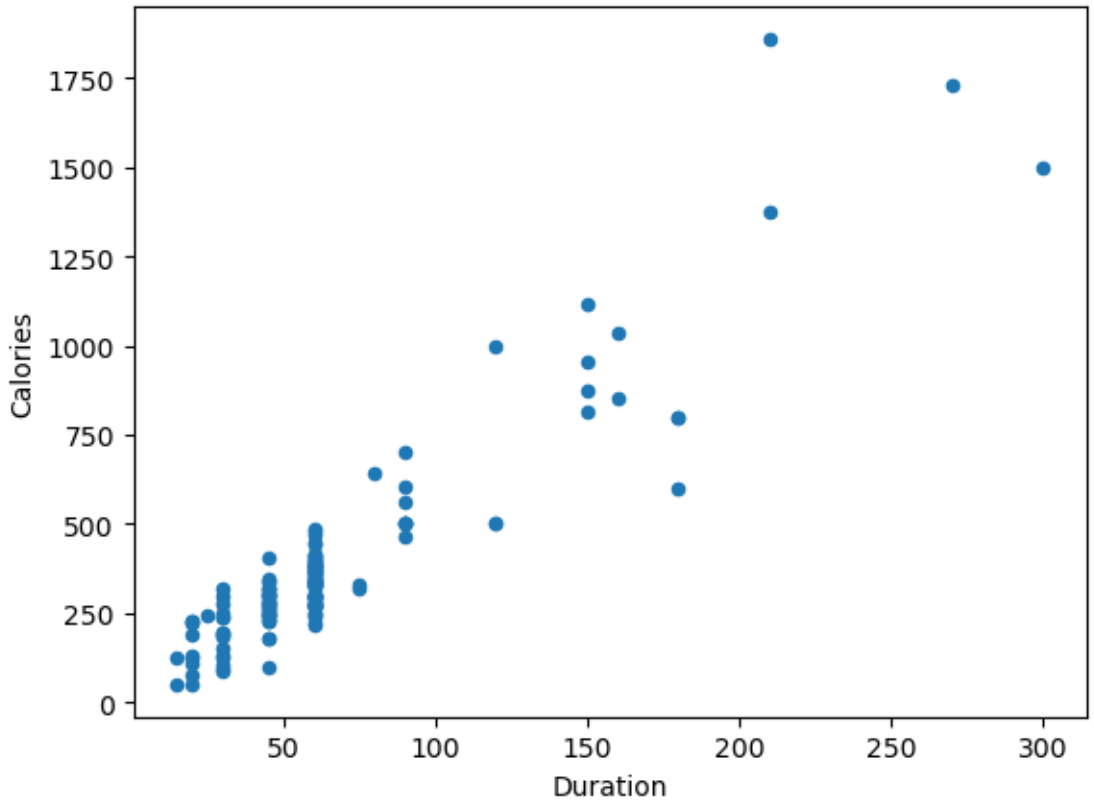
Ejemplo.

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data/data.csv')
```

```
df.plot(kind = 'scatter', x = 'Duration', y = 'Calories')

plt.show()
```



En la sección anterior, aprendimos que la correlación entre "Duración" y "Calorías" era 0.922721, y concluimos con el hecho de que una mayor duración significa más calorías quemadas.

Al observar el diagrama de dispersión, podemos observar esa relación.

Creemos otro diagrama de dispersión, donde hay una mala relación entre las columnas, como "Duración" y "Pulso máximo", con la correlación 0.009403.

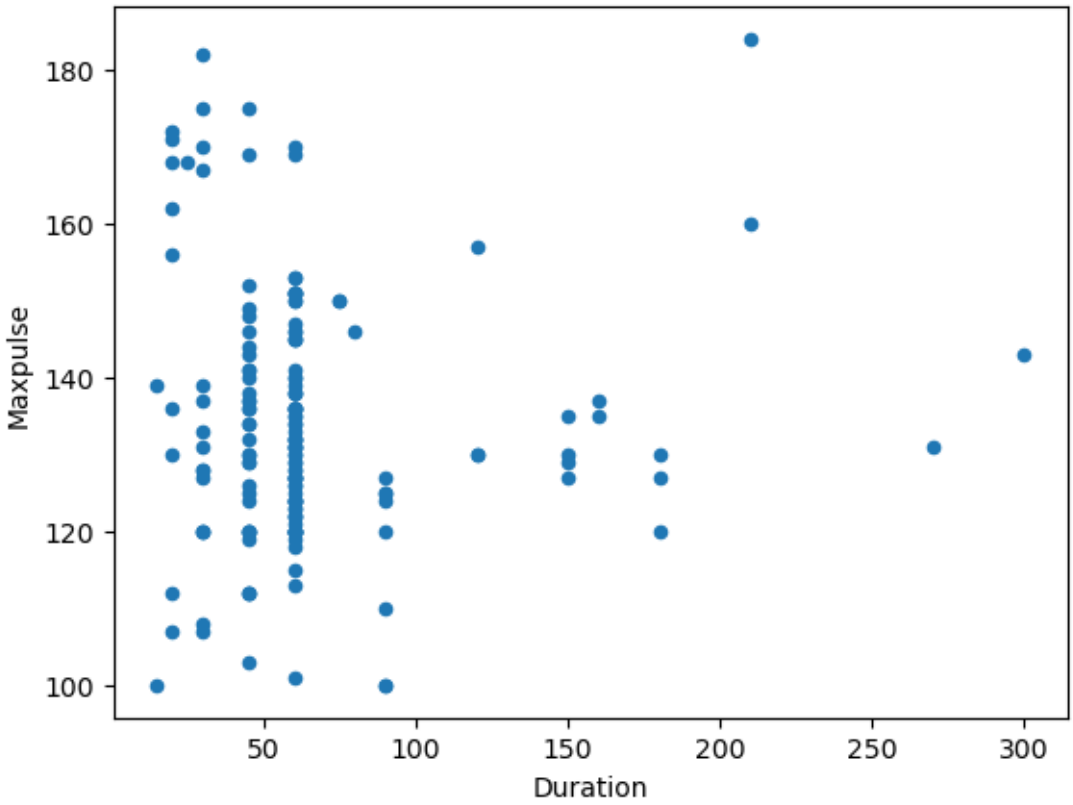
Ejemplo. Un diagrama de dispersión donde no hay relación entre las columnas.

[ ]:
```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data/data.csv')

df.plot(kind = 'scatter', x = 'Duration', y = 'Maxpulse')

plt.show()
```



4.2 Histograma

Utilice el kindargumento para especificar que desea un histograma:

```
kind = 'hist'
```

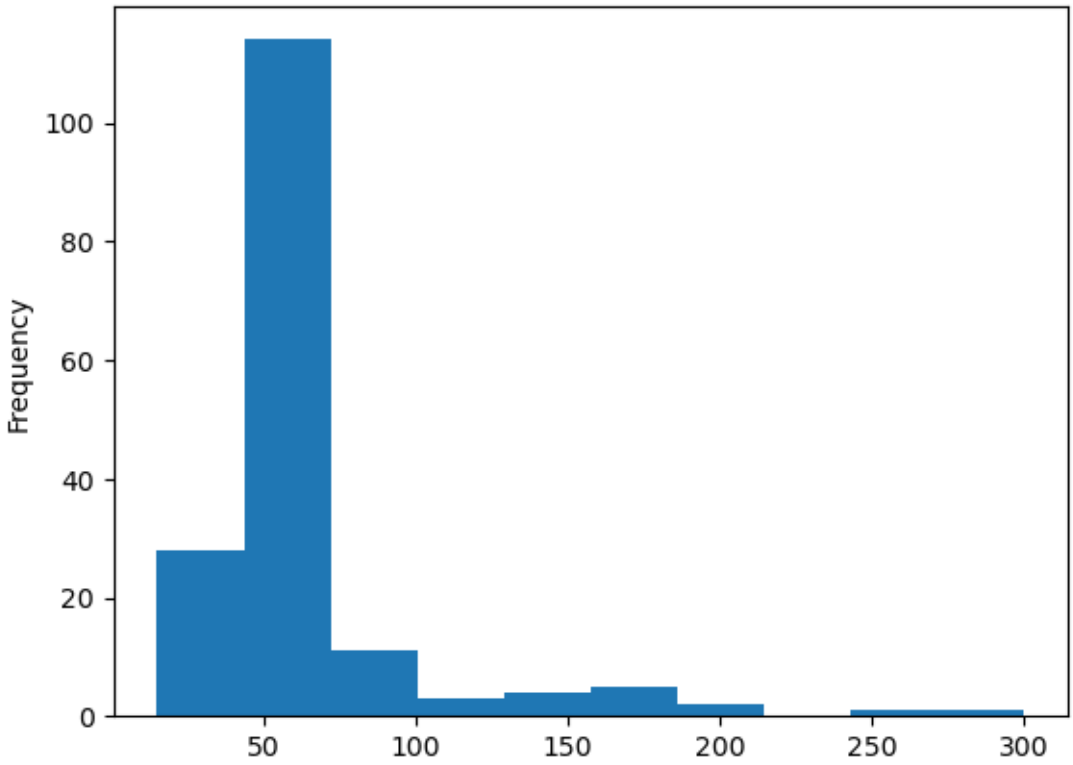
Un histograma solo necesita una columna. Un histograma nos muestra la frecuencia de cada intervalo, por ejemplo ¿cuántos entrenamientos duraron entre 50 y 60 minutos?

En el siguiente ejemplo, utilizaremos la columna "Duración" para crear el histograma.

Ejemplo

[ ]:
```
df["Duration"].plot(kind = 'hist')
```

[ ]:
```
<Axes: ylabel='Frequency'>
```

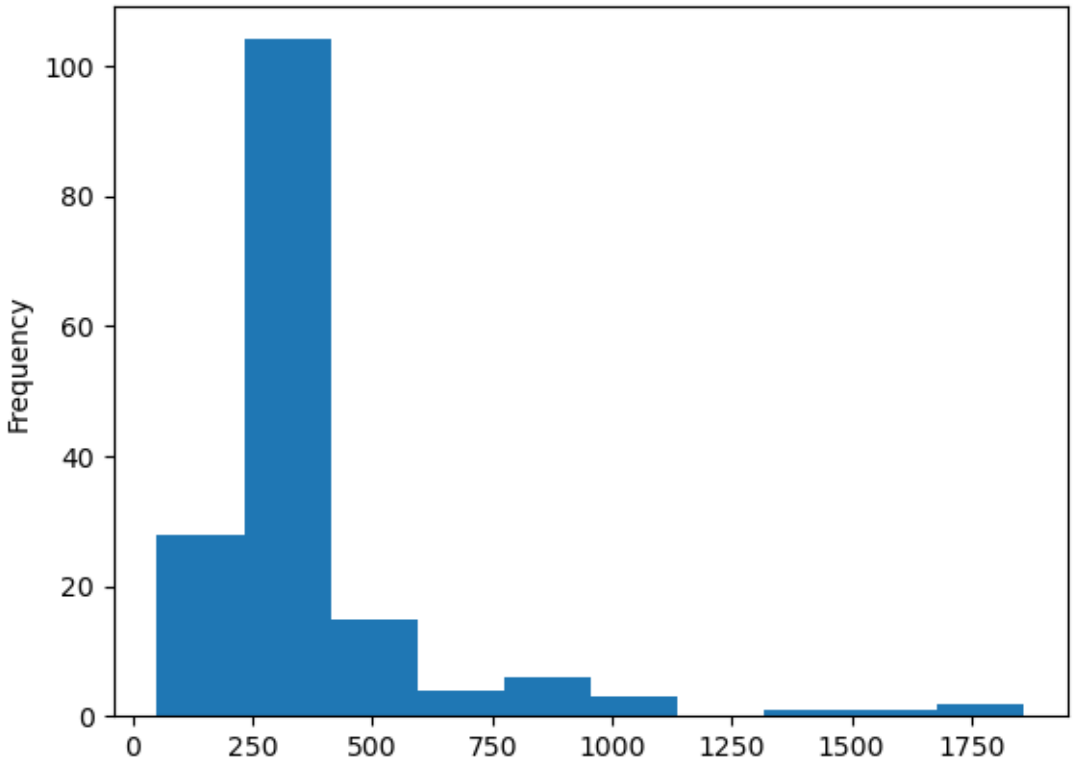


El histograma nos dice que hay más de 100 ejercicios que duraron entre 50 y 60 minutos.

Ejemplo. Ahora un histograma para las calorías.

```
[ ]: df["Calories"].plot(kind = 'hist')
```

```
[ ]: <Axes: ylabel='Frequency'>
```



Este histograma nos dice que hubo poco más de 100 casos donde las calorías quemadas fueron entre 250 y 300.

Dado que el método `plot` de Pandas utiliza `matplotlib` como backend, si se desea modificar aspectos finos de la gráfica se debe hacer configurando el gráfico a través de `matplotlib` directamente.

5 Referencias

- Manejo de archivos en Python.
- Lutz M., Learning Python, O'Reilly. 2009
- Pandas, sitio oficial.
- Pandas en la W3Schools.
- Pandas Plot