In [2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Sortir Data

In [3]:
```
import pandas as pd

data = pd.read_csv('/content/drive/MyDrive/MATERI/Pembelajaran Mesin/Praktikum Genap 20212022/data.csv')

sorted_data = data.sort_values(by=["Duration"], ascending=True)

print(sorted_data)
```

```
     Duration  Pulse  Maxpulse  Calories
112        15    124       139     124.2
93         15     80       100      50.5
135        20    136       156     189.0
68         20    106       136     110.4
64         20    110       130     131.4
..        ...    ...       ...       ...
90        180    101       127     600.1
109       210    137       184    1860.4
60        210    108       160    1376.0
79        270    100       131    1729.0
69        300    108       143    1500.2

[169 rows x 4 columns]
```

Cleaning Data

Pembersihan data berarti memperbaiki data yang buruk dalam kumpulan data Anda. Data yang buruk bisa jadi:

- Sel kosong
- Data dalam format yang salah
- Data yang salah
- Duplikat

Sel Kosong

Sel kosong berpotensi memberikan hasil yang salah saat Anda menganalisis data.

In [4]:
```
import pandas as pd

data = pd.read_csv('/content/drive/MyDrive/MATERI/Pembelajaran Mesin/Praktikum Genap 20212022/data.csv')
```

Cek Nilai Null

In [5]:
```
ceknull = data.isna()
print(ceknull)
```

```
     Duration  Pulse  Maxpulse  Calories
0       False  False     False     False
1       False  False     False     False
2       False  False     False     False
3       False  False     False     False
4       False  False     False     False
..        ...    ...       ...       ...
164     False  False     False     False
165     False  False     False     False
166     False  False     False     False
167     False  False     False     False
168     False  False     False     False

[169 rows x 4 columns]
```

Memastikan Ada Nilai Null

In [6]:
```
data.isna().values.any()
```

Out[6]:
```
True
```

Menghitung Jumlah Null Setiap Kolom

In [7]:
```
data.isna().sum()
```

Out[7]:
```
Duration    0
Pulse       0
Maxpulse    0
Calories    5
dtype: int64
```

Menampilkan Data Berdasarkan Data Null

In [8]:
```
bool_series = pd.isnull(data["Calories"])

data[bool_series]
```

Out[8]:

	Duration	Pulse	Maxpulse	Calories
17	45	90	112	NaN
27	60	103	132	NaN
91	45	107	137	NaN
118	60	105	125	NaN
141	60	97	127	NaN

Menghapus Baris

Salah satu cara untuk menangani sel kosong adalah dengan menghapus baris yang berisi sel kosong.

Ini biasanya OK, karena kumpulan data bisa sangat besar, dan menghapus beberapa baris tidak akan berdampak besar pada hasilnya.

Secara default, metode dropna () mengembalikan DataFrame baru, dan tidak akan mengubah aslinya.

Jika Anda ingin mengubah DataFrame asli, gunakan argumen inplace = True:

df.dropna(inplace = True)

print(df.to_string())

In [9]:
```
new_df = data.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
```

In [10]:
```
new_df.isna().values.any()
```

Out[10]:
```
False
```

In [11]:
```
new_df.isna().sum()
```

Out[11]:
```
Duration    0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

In [12]:
```
print("Dimensi Data Awal :", data.shape)
print("Dimensi Data Akhir :", new_df.shape)
```

```
Dimensi Data Awal : (169, 4)
Dimensi Data Akhir : (164, 4)
```

In [13]:
```
print(new_df.loc[16:18,:])
```

```
    Duration  Pulse  Maxpulse  Calories
16        60    100       120     300.0
18        60    103       123     323.0
```

Mengganti Nilai Kosong

Cara lain untuk menangani sel kosong adalah dengan memasukkan nilai baru. Dengan cara ini Anda tidak perlu menghapus seluruh baris hanya karena beberapa sel kosong. Metode fillna () memungkinkan kita mengganti sel kosong dengan nilai:

In [14]:
```
df1 = data.fillna(130)
```

In [15]:
```
df1.isna().values.any()
```

Out[15]:
```
False
```

In [16]:
```
df1.isna().sum()
```

Out[16]:
```
Duration    0
Pulse       0
Maxpulse    0
Calories    0
dtype: int64
```

In [17]:
```
print(df1.loc[16:18,:])
```

```
    Duration  Pulse  Maxpulse  Calories
16        60    100       120     300.0
17        45     90       112     130.0
18        60    103       123     323.0
```

Mengganti Hanya Untuk Kolom Tertentu

Contoh di atas menggantikan semua sel kosong di seluruh Data Frame. Untuk hanya mengganti nilai kosong untuk satu kolom, tentukan nama kolom untuk DataFrame

In [18]:
```
df2=data.copy(deep=True)
df2["Calories"].fillna(130, inplace=True)
df2.loc[16:18,:]
```

Out[18]:

	Duration	Pulse	Maxpulse	Calories
16	60	100	120	300.0
17	45	90	112	130.0
18	60	103	123	323.0

Mengganti dengan Nilai Mean, Median dan Modus

Cara umum untuk mengganti sel kosong, adalah menghitung nilai mean, median atau mode kolom. Panda menggunakan metode mean () median () dan mode () untuk menghitung nilai masing masing untuk kolom yang ditentukan:

Mean

In [19]:
```
x = data["Calories"].mean()

df3 = data.copy(deep = True)
df3["Calories"].fillna(x, inplace = True)
```

In [20]:
```
print(df3.loc[16:18,:])
```

```
    Duration  Pulse  Maxpulse    Calories
16        60    100       120  300.000000
17        45     90       112  375.790244
18        60    103       123  323.000000
```

Median

In [21]:
```
y = data["Calories"].median()
df4 = data.copy(deep = True)
df4["Calories"].fillna(y, inplace = True)
print(y)
```

```
318.6
```

In [22]:
```
print(df4.loc[16:18,:])
```

```
    Duration  Pulse  Maxpulse  Calories
16        60    100       120     300.0
17        45     90       112     318.6
18        60    103       123     323.0
```

Modus

In [23]:
```
z = data["Calories"].mode()[0]
df5 = data.copy(deep = True)
df5["Calories"].fillna(z, inplace = True)
print(z)
```

```
300.0
```

In [24]:
```
print(df5.loc[16:18,:])
```

```
    Duration  Pulse  Maxpulse  Calories
16        60    100       120     300.0
17        45     90       112     300.0
18        60    103       123     323.0
```

Data of Wrong Data

"Data salah" tidak harus berupa "sel kosong", bisa saja salah, seperti jika seseorang mendaftarkan "199", bukan "1.99".

Terkadang Anda dapat menemukan data yang salah dengan melihat kumpulan data, karena Anda memiliki ekspektasi tentang apa yang seharusnya.

Jika Anda melihat kumpulan data kami, Anda dapat melihat bahwa di baris 7, durasinya 450, tetapi untuk semua baris lainnya durasinya antara 30 dan 60.

Tidak harus salah, tetapi dengan mempertimbangkan bahwa ini adalah kumpulan data sesi latihan seseorang, kami menyimpulkan dengan fakta bahwa orang ini tidak berolahraga dalam 450 menit.

In [25]:
```
data.loc[7, 'Duration'] = 450
```

In [26]:
```
data.loc[7, 'Duration']
```

Out[26]:
```
450
```

Mengganti Nilai

In [27]:
```
data.head(10)
```

Out[27]:

	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	450	104	134	253.3
8	30	109	133	195.1
9	60	98	124	269.0

In [28]:
```
df6 = data.copy(deep = True)
df6.loc[7,'Duration'] = 45
```

In [29]:
```
df6.head(10)
```

Out[29]:

	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

Untuk kumpulan data kecil, Anda mungkin dapat mengganti data yang salah satu per satu, tetapi tidak untuk kumpulan data besar.

Untuk mengganti data yang salah untuk kumpulan data yang lebih besar, Anda dapat membuat beberapa aturan, mis. menetapkan batasan-batasan untuk nilai-nilai hukum, dan mengganti nilai-nilai yang berada di luar batasan-batasan tersebut.

In [30]:
```
df7 = data.copy(deep = True)
print(df7.loc[df7['Duration'] > 120])
```

```
     Duration  Pulse  Maxpulse  Calories
7         450    104       134     253.3
60        210    108       160    1376.0
61        160    110       137    1034.4
62        160    109       135     853.0
65        180     90       130     800.4
66        150    105       135     873.4
67        150    107       130     816.0
69        300    108       143    1500.2
70        150     97       129    1115.0
73        150     97       127     953.2
79        270    100       131    1729.0
90        180    101       127     600.1
106       180     90       120     800.3
109       210    137       184    1860.4
```

In [31]:
```
for x in df7.index:
    if df7.loc[x, "Duration"] > 120:
        df7.loc[x, "Duration"] = 120
```

In [32]:
```
print(df7.loc[df7['Duration']> 120])
```

```
Empty DataFrame
Columns: [Duration, Pulse, Maxpulse, Calories]
Index: []
```

In [33]:
```
print(df7.loc[df7['Duration']== 120])
```

```
     Duration  Pulse  Maxpulse  Calories
7         120    104       134     253.3
60        120    108       160    1376.0
61        120    110       137    1034.4
62        120    109       135     853.0
65        120     90       130     800.4
66        120    105       135     873.4
67        120    107       130     816.0
69        120    108       143    1500.2
70        120     97       129    1115.0
73        120     97       127     953.2
78        120    100       130     500.4
79        120    100       131    1729.0
83        120    100       130     500.0
87        120    100       157    1000.1
90        120    101       127     600.1
106       120     90       120     800.3
109       120    137       184    1860.4
```

Menghapus Baris

Cara lain untuk menangani data yang salah adalah dengan menghapus baris yang berisi data yang salah.

Dengan cara ini Anda tidak perlu mencari tahu apa yang harus menggantikannya, dan ada kemungkinan besar Anda tidak membutuhkan mereka untuk melakukan analisis.

In [34]:
```
df8 = data.copy(deep = True)
```

In [35]:
```
for x in df8.index:
    if df8.loc[x, "Duration"] > 120:
        df8.drop(x, inplace = True)
```

In [36]:
```
print(df8.loc[df8['Duration'] > 120])
```

```
Empty DataFrame
Columns: [Duration, Pulse, Maxpulse, Calories]
Index: []
```

Duplicate Data

Baris duplikat adalah baris yang telah didaftarkan lebih dari satu kali.

In [37]:
```
print(data.duplicated())
```

```
0      False
1      False
2      False
3      False
4      False
       ...
164    False
165    False
166    False
167    False
168    False
Length: 169, dtype: bool
```

In [38]:
```
print(data.duplicated().sum())
```

```
7
```

Menghapus Duplikat

In [39]:
```
df9 = data.copy(deep = True)
df9.drop_duplicates(inplace = True)
```

In [40]:
```
print(df9.duplicated().sum())
```

```
0
```