WS

March 17, 2025

0.1 ws_3_1

[7]:
```
import numpy as np
import pandas as pd
from scipy.stats import norm

# 1.
# -  (mu): 50
# -   (sigma): 10
# -     (size): 1000
# - numpy  random.normal( ,    ,     )
#                  .

mu, sigma, size = 50, 10, 1000
data = np.random.normal(mu, sigma, size)  #
data
```

[7]:
```
array([48.70186887, 54.64576626, 49.75454413, 41.26950162, 65.79764506,
       50.07756154, 46.39987259, 39.75944942, 47.85268102, 35.76935018,
       37.29793496, 34.20908307, 47.00547545, 43.97970305, 68.21788891,
       70.74751573, 35.57750888, 49.97189728, 56.60198186, 56.70782335,
       48.18393058, 50.24315502, 44.48694162, 63.28319248, 45.57234995,
       53.04473546, 57.31548825, 57.21230016, 48.94058308, 44.5787628 ,
       62.37934928, 53.00332751, 37.12834597, 52.42010195, 67.60777093,
       45.36639601, 60.79957809, 48.93192487, 80.26888326, 43.91583338,
       59.60146767, 48.02922145, 57.9441256 , 47.72252312, 50.73624112,
       28.17394536, 62.18132648, 61.37250207, 30.4824053 , 48.5423472 ,
       37.21734525, 47.12581909, 40.87426222, 52.41262743, 42.0659124 ,
       46.60240942, 53.32040942, 59.68448435, 57.40852497, 66.29866431,
       57.18144072, 63.15670004, 49.14763301, 46.93754507, 51.43936343,
       43.6418833 , 39.57716245, 45.48886405, 59.38370544, 66.50035784,
       41.37691142, 63.50565742, 30.86649272, 62.67492762, 19.40233766,
       62.60826514, 43.71634335, 54.20926842, 46.57146563, 33.04515337,
       53.7242119 , 33.91303161, 50.47733036, 71.50576087, 66.19264423,
       69.0229111 , 44.65597887, 52.08603786, 52.94702294, 41.59408165,
       66.17642057, 41.67295975, 59.56999448, 50.62541808, 48.30951853,
       51.51606245, 44.77823429, 44.71695375, 61.20666468, 55.57377677,
       54.28208445, 40.93997298, 38.47486983, 40.85573612, 56.13383158,
```

42.79656293, 52.62392283, 34.63327594, 52.78995927, 42.84954539,
49.13899953, 59.98782635, 53.87701133, 16.10356077, 45.74266705,
40.3447711 , 55.46226569, 53.08235352, 43.52315162, 49.99145177,
37.78309006, 40.34004368, 53.19761349, 41.49280652, 36.8717396 ,
40.06946545, 36.68265141, 49.04235251, 42.95359248, 45.32919528,
49.21886289, 46.95047459, 43.65683272, 43.69098187, 54.63614226,
56.25278732, 44.01224096, 49.14721839, 44.83051402, 51.11240204,
57.22931668, 69.08261933, 44.88027323, 45.02492697, 67.41638584,
61.92172811, 50.06221761, 48.90268502, 33.3493374 , 46.5394574 ,
43.16837239, 32.29763837, 72.26766054, 42.25146522, 41.26042484,
51.95730641, 60.00475747, 47.92886719, 52.31253151, 48.71563721,
53.03672854, 34.58808686, 52.45288051, 42.70316857, 35.76048165,
40.77201678, 44.17813617, 45.19122189, 38.13682747, 57.84746574,
27.62880583, 64.3105603 , 51.24769034, 56.85022048, 55.02177242,
40.87254917, 50.54373229, 39.56874055, 46.13748412, 50.75117183,
30.3620243 , 52.97465779, 48.2647716 , 52.35507505, 55.2163735 ,
52.23724156, 50.77416034, 45.27953818, 47.04137455, 50.02295515,
45.33706736, 54.55980895, 48.18353177, 39.18509404, 33.48131035,
51.35848558, 55.70355399, 41.74195129, 23.44542789, 42.7350022 ,
54.84860473, 52.17612025, 53.29045351, 53.33832937, 43.12556934,
46.67999552, 42.79289641, 44.18129189, 50.41417847, 53.9089427 ,
33.5515606 , 55.70164724, 45.05906442, 58.45643999, 30.14959957,
52.54464491, 58.83259893, 35.77211458, 37.57752699, 57.80750737,
36.43977579, 66.69827442, 41.25126289, 58.21232889, 46.76944304,
49.75700453, 62.87667421, 45.93432562, 37.82995202, 46.27454981,
54.59825913, 33.52945877, 50.72566169, 60.24664124, 63.34493967,
54.92270407, 51.34498701, 49.91587737, 91.33611979, 51.37980712,
69.78944196, 48.67280883, 64.67733251, 57.0556502 , 53.89555568,
40.75219927, 41.09885935, 46.03005795, 61.86389197, 44.96327273,
39.51645207, 51.68478133, 41.04172458, 60.33887106, 47.61823337,
67.05957447, 53.92087656, 58.62867091, 48.95966116, 59.86283577,
53.05207831, 55.2646306 , 39.29095374, 38.64594142, 30.67451068,
60.02406038, 39.95108745, 61.23097655, 44.02928224, 49.45663578,
53.55652505, 44.57942373, 42.14072341, 51.94789811, 49.63373095,
46.84966359, 49.64256994, 56.8160756 , 47.68345719, 56.72171818,
40.34276875, 39.99210987, 49.24443263, 50.24480463, 51.90545395,
50.41730688, 65.18161927, 56.40934276, 52.93192254, 53.62810356,
62.18565872, 52.55246538, 67.14140865, 50.31543587, 38.59036534,
60.47325432, 63.44988189, 50.54697088, 42.03976277, 61.80872219,
39.40324211, 43.80613999, 50.26153673, 47.5558496 , 45.89714027,
53.2850966 , 60.38275395, 46.55608371, 36.13428628, 49.23607924,
54.16709798, 54.81591392, 65.69230522, 37.83728352, 70.30109   ,
57.1666877 , 72.18439211, 51.64258818, 37.6348952 , 47.68592374,
57.65675526, 42.60718451, 56.78549532, 57.62218645, 40.07991255,
63.03883599, 51.50317638, 56.6132049 , 54.43100195, 47.87231795,
48.68405721, 46.58925078, 42.62644746, 59.77200156, 46.02642938,
33.79713682, 58.14118641, 68.8930628 , 42.59080752, 66.8423591 ,

55.23959344, 36.35631776, 46.99333619, 55.61421365, 68.10897825,
42.93656608, 52.73619119, 46.10574741, 43.99946133, 48.23171158,
65.49411405, 45.35240416, 55.1246865 , 63.6687694 , 61.31460343,
43.63299533, 71.57328928, 50.71247393, 41.4814215 , 43.90102462,
35.01468226, 50.14677316, 50.84222469, 51.56937339, 51.63731796,
69.28816676, 62.90025664, 51.47616313, 47.92858294, 49.98884937,
49.23010238, 56.93812893, 42.11022195, 54.98643947, 52.1025242 ,
65.60845622, 55.48263171, 50.89914749, 39.65111155, 55.1038783 ,
30.77843748, 58.818796 , 41.0368398 , 55.62080475, 52.49576858,
38.12201368, 61.83486791, 42.81840333, 39.50862353, 56.5765045 ,
73.15991929, 45.47003419, 54.64085973, 59.87060324, 26.94824813,
47.92078507, 52.83399316, 26.83913633, 67.45211044, 31.71507702,
24.20118689, 57.57942767, 57.42819223, 64.77198926, 47.68761053,
37.07640082, 32.67059078, 64.42878462, 56.85475972, 54.36123817,
59.55287914, 48.00614029, 47.01192025, 53.50227839, 63.08881982,
56.81343254, 68.14880461, 57.03551849, 43.21902586, 46.47588187,
50.6094173 , 59.22078216, 42.91983042, 61.8028074 , 52.89866493,
69.9854283 , 62.19437404, 56.78415532, 40.04495799, 31.75661448,
27.390678 , 39.25281991, 49.62928182, 65.66789493, 50.57177618,
41.52159004, 34.48745105, 32.57364558, 53.1282489 , 53.06533005,
50.44064845, 46.41522786, 17.66525237, 32.58207408, 42.75590729,
58.00271231, 42.58646096, 44.53575313, 48.44373108, 59.81146661,
46.71399469, 37.29557402, 69.54969384, 71.59554461, 58.16555058,
52.12287476, 58.39247441, 55.99759917, 51.83474927, 61.23895595,
56.11146528, 61.718313 , 55.8027121 , 40.49871383, 67.21009582,
42.89772991, 50.41977147, 43.0688685 , 49.57080042, 38.33347843,
37.60618997, 48.04462455, 31.48746469, 33.12124239, 64.30820913,
40.78409001, 63.52984395, 41.27450278, 52.26655789, 47.03267857,
35.77012141, 63.330801 , 73.88169491, 44.26949808, 36.39684224,
58.35578563, 37.25233595, 37.15099242, 33.47052693, 63.99296006,
64.18090144, 43.08627511, 70.89497963, 55.09365498, 36.63108476,
60.21662219, 54.26604663, 36.86326405, 26.41850753, 46.08803228,
41.85347255, 35.92330808, 59.40040261, 42.26255953, 39.14564544,
35.49666678, 45.53450275, 59.57701329, 55.59066386, 47.62903985,
39.58116269, 47.809979 , 60.64035004, 44.03821945, 42.06487292,
54.26903592, 38.6723983 , 40.93642394, 60.16855056, 57.47148925,
51.96637739, 58.13736743, 65.72870627, 50.42029829, 57.76988548,
52.19320403, 47.37312069, 51.70223236, 43.27063388, 33.24071538,
48.64208303, 48.15053524, 42.34967701, 58.88960943, 74.15024482,
54.03389037, 55.36322122, 55.32148924, 57.11993488, 41.58180198,
57.26893076, 69.48241674, 67.46332066, 47.62547203, 58.63156538,
59.852517 , 67.29430981, 47.55353789, 48.80598648, 68.20948107,
57.04364353, 53.30502909, 49.42967821, 69.10798291, 57.83133341,
45.79377648, 73.50236237, 61.04810553, 59.64236921, 43.53233079,
58.4274324 , 48.70370525, 46.6245553 , 40.73738716, 52.98687619,
56.35301793, 31.98307568, 59.32056621, 46.43650219, 44.12990348,
40.06812068, 37.59664603, 61.70654165, 43.05222256, 50.25662496,

```
54.88796652, 35.67094059, 55.24883665, 48.9398363 , 39.6484908 ,
37.97942414, 48.24905602, 56.92612619, 40.95927985, 34.40853042,
53.48364285, 50.30336788, 49.65612843, 36.13825781, 67.15963586,
57.85149737, 31.00540226, 31.64881083, 51.03366573, 48.09975582,
54.19458826, 61.58158037, 59.57758225, 57.11865005, 62.22688539,
42.21062605, 67.5091989 , 51.33454807, 66.47834966, 40.20029074,
67.1676658 , 51.23569873, 43.44002522, 50.75343676, 55.61235954,
58.89784679, 46.37830641, 42.55602518, 69.46618671, 59.33898192,
54.61771918, 38.00566301, 52.49713946, 37.75690862, 29.96753929,
35.47761713, 40.77833011, 68.95218836, 70.85996868, 63.57897648,
41.14213756, 61.85296855, 34.66066232, 44.78688203, 63.43554081,
42.99131666, 51.43324622, 57.39611108, 45.61610442, 62.52794026,
63.4979273 , 43.08258011, 49.38907478, 75.51644404, 50.4360237 ,
51.68987398, 44.80475333, 35.2756896 , 56.68783837, 25.8729816 ,
47.1825976 , 41.70171918, 44.28417931, 33.52637805, 45.76401363,
41.55655282, 50.41610264, 41.28525117, 39.57003263, 49.67582377,
44.3710939 , 63.08365753, 54.5130913 , 64.09701445, 40.78539006,
46.13013541, 57.34835729, 48.14081836, 44.6867945 , 52.48010832,
58.99702102, 51.15614662, 51.42019401, 64.16422213, 48.73825237,
54.05735503, 52.01113213, 51.94663104, 47.22687139, 49.58683829,
54.18698737, 37.16444916, 50.36286462, 51.59291252, 64.80793457,
45.12129338, 47.5463139 , 38.37354716, 38.99673574, 48.99514771,
56.48125049, 42.04766753, 46.96085876, 43.71684243, 46.19272103,
55.32485307, 37.90565948, 54.40263664, 57.66410572, 55.27428949,
55.72596299, 58.84077323, 58.03787643, 73.48407713, 46.63278995,
61.33677676, 52.38445876, 44.93514675, 57.06432123, 29.06898852,
57.62308675, 63.33023356, 37.38142629, 46.04431641, 54.7625017 ,
54.95811312, 69.56566788, 46.93095908, 52.20757896, 39.13775932,
52.80011848, 78.4896489 , 59.37677666, 44.46586068, 47.46962745,
55.41014245, 35.30281633, 44.1635289 , 37.90563128, 50.54744861,
43.93643239, 66.33547785, 47.04747103, 45.19230909, 45.23883046,
48.76447589, 31.6993952 , 50.93116052, 54.48585163, 40.71375184,
65.75146413, 48.85789112, 44.8597926 , 62.29708775, 42.71957378,
74.63184774, 51.46563306, 48.34034579, 37.14718448, 65.3573946 ,
50.57994124, 47.18673733, 39.63700227, 58.37353247, 57.03613581,
54.34675089, 42.0251978 , 40.35951563, 36.19117703, 52.88072234,
57.01361955, 50.68178759, 33.50372975, 51.242185  , 44.49725483,
49.67102483, 44.44843402, 51.50432069, 54.01120468, 59.67097388,
38.71076716, 36.68258164, 47.53955402, 58.80807582, 53.79659884,
46.29600042, 58.45671857, 28.33843381, 26.02249626, 46.49942697,
43.04189102, 43.27323538, 51.33702868, 43.90939392, 46.84170164,
51.07670899, 46.8644695 , 62.25859979, 28.53624954, 56.09599172,
54.69423855, 65.36851545, 41.68062102, 55.41266966, 53.52181442,
46.75910003, 50.24701792, 42.82320785, 55.25542765, 44.99710708,
32.23616895, 48.6686769 , 44.98736724, 44.39450991, 66.3836836 ,
59.95787505, 49.22786585, 23.05425039, 54.79885629, 50.77748222,
32.71005462, 33.41366641, 42.6757654 , 50.55107365, 43.96559096,
```

```
       47.71255094, 58.7415243 , 66.23679418, 48.55098855, 43.37893783,
       49.83931483, 47.59176279, 48.44124496, 59.25571519, 38.97629566,
       52.18868664, 48.85609823, 53.29520231, 34.64789451, 43.13360993,
       46.70778323, 45.23722024, 43.99049044, 40.99453605, 48.44948334,
       51.63363803, 54.20919807, 46.00544485, 43.93178607, 42.71687121,
       31.19251815, 57.8482993 , 61.03225736, 36.61339543, 54.65287184,
       44.47203281, 57.16708056, 50.27913631, 53.77352338, 51.44333071,
       41.68100054, 50.15367041, 20.92387369, 63.37608191, 53.21680732,
       55.00739483, 44.61429562, 52.94672427, 39.28909345, 57.89041571,
       56.13034024, 49.39480032, 49.82600579, 52.23818985, 54.92449139,
       58.8456482 , 43.00876705, 32.7498294 , 45.61194606, 40.70302458,
       62.33503555, 40.55565723, 41.15743875, 33.09170862, 34.23814999,
       48.25901464, 55.15336744, 55.22172958, 47.80255845, 34.73936645,
       63.79500211, 42.88883931, 40.31499202, 54.05432091, 48.98688953,
       58.48250641, 35.9739528 , 41.88576043, 49.3317393 , 60.05286817,
       65.9325587 , 46.86994737, 65.40897507, 60.84856452, 43.34614254,
       49.2053298 , 40.46499507, 59.762311  , 52.75329019, 48.93839614,
       62.04452816, 54.67338197, 73.44972213, 65.27168879, 50.4948676 ,
       36.40911987, 56.48239395, 46.33050792, 44.17415458, 59.01689667,
       47.61637421, 42.72601279, 44.11985506, 57.75321691, 63.78718679,
       50.98069108, 45.41764649, 45.71224039, 56.43806923, 26.28335613,
       57.52873155, 44.59482941, 55.34202392, 55.45855823, 57.72442879,
       68.09885999, 51.55995616, 43.56025387, 47.79748324, 56.79419078,
       56.59711889, 48.87069646, 40.49431953, 55.90884038, 34.32628326,
       44.87528527, 64.60680585, 59.36163265, 66.69303433, 61.94940754,
       48.52601321, 54.36287159, 53.90004272, 63.17484981, 50.26967534,
       46.81637594, 56.71439517, 43.12585317, 34.33813935, 56.23446745,
       64.37975415, 45.76370214, 52.49290714, 55.15743645, 71.91506928,
       31.71831768, 63.40456121, 60.56762736, 61.2132139 , 41.72714634,
       47.69119256, 31.51864079, 57.27247584, 39.24667848, 40.39012898,
       52.47491059, 54.41161558, 58.3260719 , 36.68295332, 48.47006488,
       60.1694679 , 57.97050228, 68.22010273, 44.90326519, 32.28910101,
       57.54345323, 61.78178692, 38.66870043, 51.46222761, 38.22357006,
       49.87547259, 50.76591103, 61.08037575, 56.95292215, 57.77192985,
       38.87340562, 57.97872424, 48.66488058, 45.95842591, 37.61535613,
       42.86380588, 75.11616543, 53.87915565, 38.39426674, 53.80068445,
       55.66354968, 54.67271562, 50.95562924, 48.8667915 , 51.87912183,
       48.52018934, 39.06303376, 61.39219352, 63.80450565, 54.07042384])
```

[8]:
```
# 2.    DataFrame
# - pandas  DataFrame()          .
df = pd.DataFrame({"value": data})  #
df
```

[8]:

	value
0	48.701869
1	54.645766

```
2    49.754544
3    41.269502
4    65.797645
..         …
995  48.520189
996  39.063034
997  61.392194
998  63.804506
999  54.070424

[1000 rows x 1 columns]
```

[9]:
```
# 3.
# - describe()        ,   ,   ,           .
summary_stats = df["value"].describe()  #      describe()
summary_stats
```

[9]:
```
count    1000.000000
mean       50.054848
std        10.153620
min        16.103561
25%        43.085351
50%        50.150222
75%        56.941827
max        91.336120
Name: value, dtype: float64
```

[11]:
```
# 4.
# -      (mu)    (sigma)
# - 68-95-99.7

#
# -              .
mean_value = df["value"].mean()  #
std_value = df["value"].std()  #

# 68-95-99.7
# -
# -  (mu) ±1    (sigma)     68%
# -   ±2         95%
# -   ±3         99.7%

within_1_sigma = np.sum((df["value"] >= mu - sigma) & (df["value"] <= mu +␣
 ↪sigma)) / size * 100  # ±1
within_2_sigma = np.sum((df["value"] >= mu - 2*sigma) & (df["value"] <= mu +␣
 ↪2*sigma)) / size * 100  # ±2
```

```
within_3_sigma = np.sum((df["value"] >= mu - 3*sigma) & (df["value"] <= mu +␣
↪3*sigma)) / size * 100  # ±3

print(within_1_sigma)
print(within_2_sigma)
print(within_3_sigma)
```

```
68.7
95.8
99.5
```

[12]:
```
# 5.
# -
# -    ( :  )      (PDF)
# - pdf(   ,   ,    ) =>

sample_value = mu  #
pdf_value = norm.pdf(sample_value, loc=mu, scale=sigma)  #      PDF
print(pdf_value)
```

```
0.03989422804014327
```

[13]:
```
# 6.
# -         50,    10
# - 68-95-99.7       ,
# -
# -        ( : QQ-Plot,   -     )

print("         :")
print(summary_stats)

print("\n68-95-99.7       :")
print(f"68%         : {within_1_sigma:.2f}%")
print(f"95%         : {within_2_sigma:.2f}%")
print(f"99.7%         : {within_3_sigma:.2f}%")

print("\n       :")
print(f"    : {mean_value:.2f},      : {std_value:.2f}")
print(f"        : {mu},    : {sigma}")
print(f"   ({sample_value})         (PDF)  : {pdf_value:.6f}")
```

```
         :
count    1000.000000
mean       50.054848
std        10.153620
min        16.103561
25%        43.085351
50%        50.150222
75%        56.941827
```

```
max         91.336120
Name: value, dtype: float64

68-95-99.7       :
68%          : 68.70%
95%          : 95.80%
99.7%          : 99.50%

       :
    : 50.05,      : 10.15
        : 50,    : 10
   (50)          (PDF)  : 0.039894
```

0.2 ws_3_2

[15]:
```
#
import pandas as pd

# 1. CSV
# - pandas  read_csv()             .
file_path = "sample_data.csv"
df = pd.read_csv(file_path)
df
```

[15]:

	column_name	other_column_name
0	52	37.847975
1	93	98.954011
2	15	-5.791277
3	72	71.785250
4	61	54.829096
..	…	…
995	42	45.034448
996	41	48.342609
997	6	-3.933782
998	52	51.800246
999	26	19.334795

[1000 rows x 2 columns]

[16]:
```
# 2.        ( ,     ,      )
# - info()                .
# - df.info()     None
# -             .
# -  ,     df.info()
#   print(df_info)
#   info   None
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   column_name        1000 non-null   int64  
 1   other_column_name  1000 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 15.8 KB
```

[17]:
```
# 3.
# - describe()           ,   ,  /      .
summary_stats = df.describe()
summary_stats
```

[17]:

	column_name	other_column_name
count	1000.000000	1000.000000
mean	49.560000	50.321447
std	29.287679	31.288151
min	1.000000	-19.155994
25%	24.000000	25.246078
50%	50.000000	50.325105
75%	75.000000	76.462954
max	99.000000	120.133939

[18]:
```
# 4.
# - column_name  50                (filtered_df)     .
filtered_df = df[df["column_name"] >= 50]
filtered_df
```

[18]:

	column_name	other_column_name
0	52	37.847975
1	93	98.954011
3	72	71.785250
4	61	54.829096
6	83	94.888010
..	…	…
986	68	63.418630
988	67	63.060186
990	86	71.554426
994	83	77.623103
998	52	51.800246

[505 rows x 2 columns]

[19]:
```
# - other_column_name  0                (positive_df)     .
positive_df = df[df["other_column_name"] > 0]
positive_df
```

[19]:
```
     column_name  other_column_name
0             52          37.847975
1             93          98.954011
3             72          71.785250
4             61          54.829096
5             21           8.411580
..           …                 …
994           83          77.623103
995           42          45.034448
996           41          48.342609
998           52          51.800246
999           26          19.334795

[955 rows x 2 columns]
```

[20]:
```
# 5.
# - column_name  50    , other_column_name  0           .
filtered_combined = df[(df["column_name"] >= 50) & (df["other_column_name"] >␣
↪0)]
filtered_combined
```

[20]:
```
     column_name  other_column_name
0             52          37.847975
1             93          98.954011
3             72          71.785250
4             61          54.829096
6             83          94.888010
..           …                 …
986           68          63.418630
988           67          63.060186
990           86          71.554426
994           83          77.623103
998           52          51.800246

[505 rows x 2 columns]
```

[21]:
```
# 6.
# - corr()       column_name  other_column_name          .
correlation = df[['column_name','other_column_name']].corr()
correlation
```

[21]:
```
                    column_name  other_column_name
column_name             1.00000            0.94231
other_column_name       0.94231            1.00000
```

[22]:
```
# 7.
print("     :")
```

```
print(df.info())  #     ,   ,     ,

print("\n           :")
print(summary_stats)  #  ,    ,   ,

print("\n  column_name  50       :", len(filtered_df))
#    :
# -      1000    column_name  50           .

print("\n  other_column_name  0       :", len(positive_df))
#    :
# -         other_column_name  0           .

print("\n      (column_name >= 50 & other_column_name > 0)        :",␣
 ↪len(filtered_combined))
#    :
# -                    .

print("\n       :")
print(correlation)
#    :
# - column_name  other_column_name        .

# 8.
# -    (column_name >= 50, other_column_name > 0)                .
# - describe()                    .
# -      (column_name  other_column_name)             (0.94)    .
# -                                 .
```

```
     :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   column_name        1000 non-null   int64
 1   other_column_name  1000 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 15.8 KB
None

          :
       column_name  other_column_name
count  1000.000000        1000.000000
mean     49.560000          50.321447
std      29.287679          31.288151
min       1.000000         -19.155994
25%      24.000000          25.246078
```

```
50%       50.000000          50.325105
75%       75.000000          76.462954
max       99.000000         120.133939

 column_name  50       : 505

 other_column_name  0       : 955

      (column_name >= 50 & other_column_name > 0)        : 505

       :
                     column_name  other_column_name
column_name              1.00000            0.94231
other_column_name        0.94231            1.00000
```

0.3 ws_3_3

[32]:
```
#
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#      AppleGothic
plt.rcParams['font.family'] = 'AppleGothic'
#  (-)           (    )
plt.rcParams['axes.unicode_minus'] = False


# 1. CSV
# - pandas  read_csv()             .
file_path = "statistics_data.csv"
df = pd.read_csv(file_path)
df
```

[32]:

	column_1	column_2	column_3	column_4
0	54.967142	36.996777	30	-0.125454
1	48.617357	34.623168	95	0.327880
2	56.476885	30.298152	75	0.085893
3	65.230299	26.765316	99	-2.219300
4	47.658466	33.491117	72	-0.229800
..	…	…	…	…
995	47.188997	35.350751	25	-0.533600
996	67.976865	29.867394	55	1.220821
997	56.408429	25.590627	46	-0.876774
998	44.288210	29.184665	32	1.712040
999	55.725828	26.275487	55	-1.747637

```
[1000 rows x 4 columns]
```

[33]:
```
# 2.        (  ,     ,      )
# - info()                .
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   column_1  1000 non-null   float64
 1   column_2  1000 non-null   float64
 2   column_3  1000 non-null   int64
 3   column_4  1000 non-null   float64
dtypes: float64(3), int64(1)
memory usage: 31.4 KB
```

[34]:
```
# 3.
# -          ,                .

#
columns_to_analyze = df.columns
columns_to_analyze
```

[34]:
```
Index(['column_1', 'column_2', 'column_3', 'column_4'], dtype='object')
```

[35]:
```
# 4.        (Quartiles), IQR

#
boxplot_stats = {}

for column in columns_to_analyze:
    # Q1 ( 1   , 25%)
    Q1 = df[column].quantile(.25)

    # Q2 (  , 50%)
    Q2 = df[column].median()

    # Q3 ( 3   , 75%)
    Q3 = df[column].quantile(.75)

    # IQR (    , Interquartile Range)
    IQR = Q3 - Q1

    #        (1.5 * IQR   3.0 * IQR    )
    lower_bound_1_5 = Q1 - 1.5 * IQR  #
    upper_bound_1_5 = Q3 + 1.5 * IQR  #
```

```
    lower_bound_3_0 = Q1 - 3.0 * IQR  #
    upper_bound_3_0 = Q3 + 3.0 * IQR  #

    #        (1.5 * IQR  )
    mild_outliers = df[(df[column] < lower_bound_1_5) | (df[column] >␣
 ↪upper_bound_1_5)]

    #         (3.0 * IQR  )
    extreme_outliers = df[(df[column] < lower_bound_3_0) | (df[column] >␣
 ↪upper_bound_3_0)]

    #
    boxplot_stats[column] = {
        "Q1": Q1,
        "Q2 (Median)": Q2,
        "Q3": Q3,
        "IQR": IQR,
        "Lower Bound (1.5 * IQR)": lower_bound_1_5,
        "Upper Bound (1.5 * IQR)": upper_bound_1_5,
        "Lower Bound (3.0 * IQR)": lower_bound_3_0,
        "Upper Bound (3.0 * IQR)": upper_bound_3_0,
        "Mild Outliers Count (1.5 * IQR)": len(mild_outliers),
        "Extreme Outliers Count (3.0 * IQR)": len(extreme_outliers),
        "Variance": np.var(df[column]),
        "Standard Deviation": np.std(df[column]),
        "Mean Absolute Deviation": np.mean(np.abs(df[column] - df[column].
 ↪median()))
    }
```

[36]:
```
# 5.
for column, stats in boxplot_stats.items():
    print(f"\n{column}                :")
    print(f"  - 1  (Q1, 25%): {stats['Q1']:.2f}")
    print(f"  -   (Q2, 50%): {stats['Q2 (Median)']:.2f}")
    print(f"  - 3  (Q3, 75%): {stats['Q3']:.2f}")
    print(f"  - IQR (Q3 - Q1): {stats['IQR']:.2f}")
    print(f"  -        (1.5 * IQR): {stats['Lower Bound (1.5 * IQR)']:.2f}")
    print(f"  -        (1.5 * IQR): {stats['Upper Bound (1.5 * IQR)']:.2f}")
    print(f"  -        (3.0 * IQR): {stats['Lower Bound (3.0 * IQR)']:.2f}")
    print(f"  -        (3.0 * IQR): {stats['Upper Bound (3.0 * IQR)']:.2f}")
    print(f"  -        (1.5 * IQR  ): {stats['Mild Outliers Count (1.5 *␣
 ↪IQR)']} ")
    print(f"  -        (3.0 * IQR  ): {stats['Extreme Outliers Count (3.0 *␣
 ↪IQR)']} ")
    print(f"  -   (Variance): {stats['Variance']:.2f}")
    print(f"  -    (Standard Deviation): {stats['Standard Deviation']:.2f}")
```

```
    print(f"  -       (Mean Absolute Deviation): {stats['Mean Absolute␣
↪Deviation']:.2f}")
```

```
column_1              :
  - 1  (Q1, 25%): 43.52
  -  (Q2, 50%): 50.25
  - 3  (Q3, 75%): 56.48
  - IQR (Q3 - Q1): 12.96
  -        (1.5 * IQR): 24.09
  -        (1.5 * IQR): 75.91
  -        (3.0 * IQR): 4.66
  -        (3.0 * IQR): 95.35
  -        (1.5 * IQR  ): 8
  -        (3.0 * IQR  ): 0
  -   (Variance): 95.79
  -    (Standard Deviation): 9.79
  -       (Mean Absolute Deviation): 7.79

column_2              :
  - 1  (Q1, 25%): 26.97
  -  (Q2, 50%): 30.32
  - 3  (Q3, 75%): 33.64
  - IQR (Q3 - Q1): 6.68
  -        (1.5 * IQR): 16.96
  -        (1.5 * IQR): 43.66
  -        (3.0 * IQR): 6.94
  -        (3.0 * IQR): 53.67
  -        (1.5 * IQR  ): 8
  -        (3.0 * IQR  ): 0
  -   (Variance): 24.85
  -    (Standard Deviation): 4.98
  -       (Mean Absolute Deviation): 3.95

column_3              :
  - 1  (Q1, 25%): 26.00
  -  (Q2, 50%): 51.50
  - 3  (Q3, 75%): 75.00
  - IQR (Q3 - Q1): 49.00
  -        (1.5 * IQR): -47.50
  -        (1.5 * IQR): 148.50
  -        (3.0 * IQR): -121.00
  -        (3.0 * IQR): 222.00
  -        (1.5 * IQR  ): 0
  -        (3.0 * IQR  ): 0
  -   (Variance): 791.33
  -    (Standard Deviation): 28.13
  -       (Mean Absolute Deviation): 24.28
```

```
column_4               :
  - 1  (Q1, 25%): -0.70
  -  (Q2, 50%): -0.00
  - 3  (Q3, 75%): 0.67
  - IQR (Q3 - Q1): 1.37
  -        (1.5 * IQR): -2.75
  -        (1.5 * IQR): 2.72
  -        (3.0 * IQR): -4.80
  -        (3.0 * IQR): 4.77
  -        (1.5 * IQR  ): 7
  -        (3.0 * IQR  ): 0
  -   (Variance): 1.04
  -    (Standard Deviation): 1.02
  -       (Mean Absolute Deviation): 0.82
```

[37]:
```
# 6.
print("\n         :")
df.describe()
```

         :

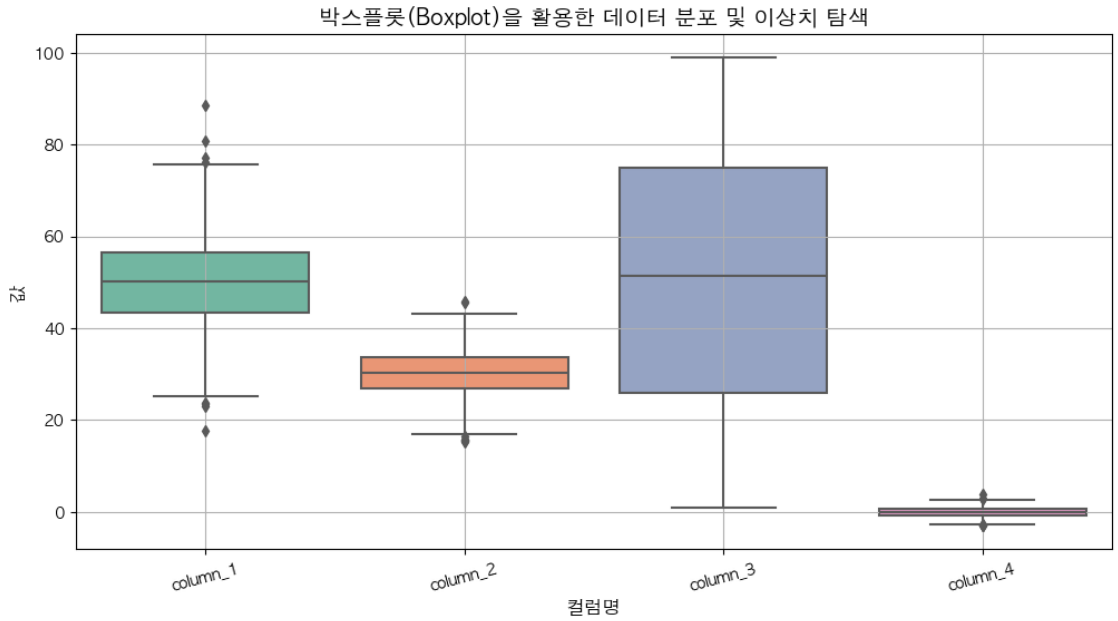
[37]:

	column_1	column_2	column_3	column_4
count	1000.000000	1000.000000	1000.000000	1000.000000
mean	50.193321	30.354181	50.465000	0.000114
std	9.792159	4.987272	28.144678	1.020034
min	17.587327	15.298057	1.000000	-2.991136
25%	43.524097	26.968792	26.000000	-0.699302
50%	50.253006	30.315386	51.500000	-0.001359
75%	56.479439	33.644411	75.000000	0.668419
max	88.527315	45.965538	99.000000	3.926238

[38]:
```
# 7.
# -                          .
plt.figure(figsize=(12, 6))  #
sns.boxplot(data=df, palette="Set2")  # seaborn
plt.title("  (Boxplot)                ", fontsize=14)
plt.xlabel("  ", fontsize=12)
plt.ylabel(" ", fontsize=12)
plt.xticks(rotation=15)  # x
plt.grid(True)  #
plt.show()


# 8.
# -        (Q1, Q2, Q3, IQR)              .
# -      (1.5 * IQR)       (3.0 * IQR)       .
```

```
# -                    .
# -      (MAD)                    ,
#              ,                  .
# -            ,              ( ,   ,    )        .
```



0.4 ws_3_4

[39]:
```
#
import pandas as pd

# 1. CSV
# - pandas  read_csv()              .
file_path = "preprocessing_data.csv"
df = pd.read_csv(file_path)
df
```

[39]:

	column_1	column_2	column_3	column_with_missing
0	54.967142	36.996777	30	19.623637
1	48.617357	34.623168	95	20.983639
2	56.476885	30.298152	75	20.257679
3	65.230299	26.765316	99	13.342099
4	47.658466	33.491117	72	NaN
..	…	…	…	…
995	47.188997	35.350751	25	18.399199
996	67.976865	29.867394	55	NaN
997	56.408429	25.590627	46	17.369679

```
998  44.288210  29.184665        32            25.136121
999  55.725828  26.275487        55                  NaN

[1000 rows x 4 columns]
```

[40]:
```
# 2.        (  ,     ,      )
# - info()               .
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   column_1             1000 non-null   float64
 1   column_2             1000 non-null   float64
 2   column_3             1000 non-null   int64
 3   column_with_missing  900 non-null    float64
dtypes: float64(3), int64(1)
memory usage: 31.4 KB
```

[41]:
```
# 3.
# - isnull().sum()                  .
missing_values = df.isnull().sum()
missing_values
```

[41]:
```
column_1                0
column_2                0
column_3                0
column_with_missing    100
dtype: int64
```

[42]:
```
# 4.
# -    (column_with_missing)       (median)     .
df['column_with_missing'] = df["column_with_missing"].
  ↪fillna(df["column_with_missing"].median())
missing_values_new = df.isnull().sum()
missing_values_new
```

[42]:
```
column_1               0
column_2               0
column_3               0
column_with_missing    0
dtype: int64
```

[43]:
```
# 5.
# - column_1  column_2            'column_mean'     .
df["column_mean"] = (df["column_1"] + df["column_2"]) / 2
```

```
# - column_3        /          'is_even'     .
df["is_even"] = df["column_3"] % 2 == 0  # True    , False

df
```

[43]:

	column_1	column_2	column_3	column_with_missing	column_mean	is_even
0	54.967142	36.996777	30	19.623637	45.981959	True
1	48.617357	34.623168	95	20.983639	41.620263	False
2	56.476885	30.298152	75	20.257679	43.387519	False
3	65.230299	26.765316	99	13.342099	45.997807	False
4	47.658466	33.491117	72	20.051030	40.574791	True
..	…	…	…	…	…	…
995	47.188997	35.350751	25	18.399199	41.269874	False
996	67.976865	29.867394	55	20.051030	48.922129	False
997	56.408429	25.590627	46	17.369679	40.999528	True
998	44.288210	29.184665	32	25.136121	36.736438	True
999	55.725828	26.275487	55	20.051030	41.000657	False

[1000 rows x 6 columns]

[44]:
```
# 6.
print("     :")
print(df.info())  #

print("\n        :")
print(df.isnull().sum())  #

print("\n            :")
print(df[['column_1', 'column_2', 'column_mean', 'column_3', 'is_even']].
 ↪head())  #   5

# 7.
# - 'column_with_missing'        (median)    .
# - column_1  column_2     'column_mean'                    .
# - column_3        /      'is_even'                  .
# -                                .
```

```
     :
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   column_1             1000 non-null   float64
 1   column_2             1000 non-null   float64
 2   column_3             1000 non-null   int64
 3   column_with_missing  1000 non-null   float64
```

```
 4   column_mean          1000 non-null   float64
 5   is_even              1000 non-null   bool
dtypes: bool(1), float64(4), int64(1)
memory usage: 40.2 KB
None
```

```
       :
column_1               0
column_2               0
column_3               0
column_with_missing    0
column_mean            0
is_even                0
dtype: int64
```

```
            :
    column_1   column_2  column_mean  column_3  is_even
0  54.967142  36.996777    45.981959        30     True
1  48.617357  34.623168    41.620263        95    False
2  56.476885  30.298152    43.387519        75    False
3  65.230299  26.765316    45.997807        99    False
4  47.658466  33.491117    40.574791        72     True
```

0.5 ws_3_5

[1]:
```
#
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
#      AppleGothic
plt.rcParams['font.family'] = 'AppleGothic'
#  (-)           (    )
plt.rcParams['axes.unicode_minus'] = False

# 1. CSV
file_path = "eda_data.csv"
df = pd.read_csv(file_path)
df
```

[1]:

	Date	column_1	column_2	column_3	column_4
0	2023-01-01	104.967142	191.975591	305.875358	396.771337
1	2023-01-02	98.617357	204.481850	270.648817	403.145408
2	2023-01-03	106.476885	200.251848	312.247583	320.071973
3	2023-01-04	115.230299	201.953522	248.922492	436.653107
4	2023-01-05	97.658466	184.539804	330.874669	413.859539
..	…	…	…	…	…
360	2023-12-27	105.193465	196.320333	297.851962	329.569490

```
361  2023-12-28  115.327389  200.368679  298.883333  430.133665
362  2023-12-29   98.912399  206.951634  321.828886  415.246335
363  2023-12-30  104.017117  189.204806  301.558377  451.590110
364  2023-12-31  106.901440  184.433905  321.979202  426.927254

[365 rows x 5 columns]
```

[2]:
```
print('## Old value ##')
print()
print(df.dtypes)
print()
print(df.isnull().sum())
print()

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

#          (            )
#
df = df.fillna(df.mean())  #

print('## New Value ##')
print()
print(df.dtypes)
print()
print(df.isnull().sum())
```

```
## Old value ##

Date         object
column_1    float64
column_2    float64
column_3    float64
column_4    float64
dtype: object

Date        0
column_1    0
column_2    0
column_3    0
column_4    0
dtype: int64

## New Value ##

Date        datetime64[ns]
column_1           float64
column_2           float64
```

```
column_3           float64
column_4           float64
dtype: object

Date        0
column_1    0
column_2    0
column_3    0
column_4    0
dtype: int64
```

[3]:
```
#          (         IQR     )
def remove_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(.25)  # 1     (Q1)
        Q3 = df[col].quantile(.75)  # 3     (Q3)
        IQR = Q3 - Q1  #      (Interquartile Range, IQR)
        lower_bound = Q1 - 1.5 * IQR  #
        upper_bound = Q3 + 1.5 * IQR  #
        #
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]  #
    return df

#
df = remove_outliers(df, ['column_1', 'column_2', 'column_3', 'column_4'])  # ␣
↪

#
# count( ), mean( ), std(  ), min(  ), max(  )
descriptive_stats = df.dtypes  #
print("    :")
descriptive_stats
```

    :

[3]:
```
Date        datetime64[ns]
column_1           float64
column_2           float64
column_3           float64
column_4           float64
dtype: object
```

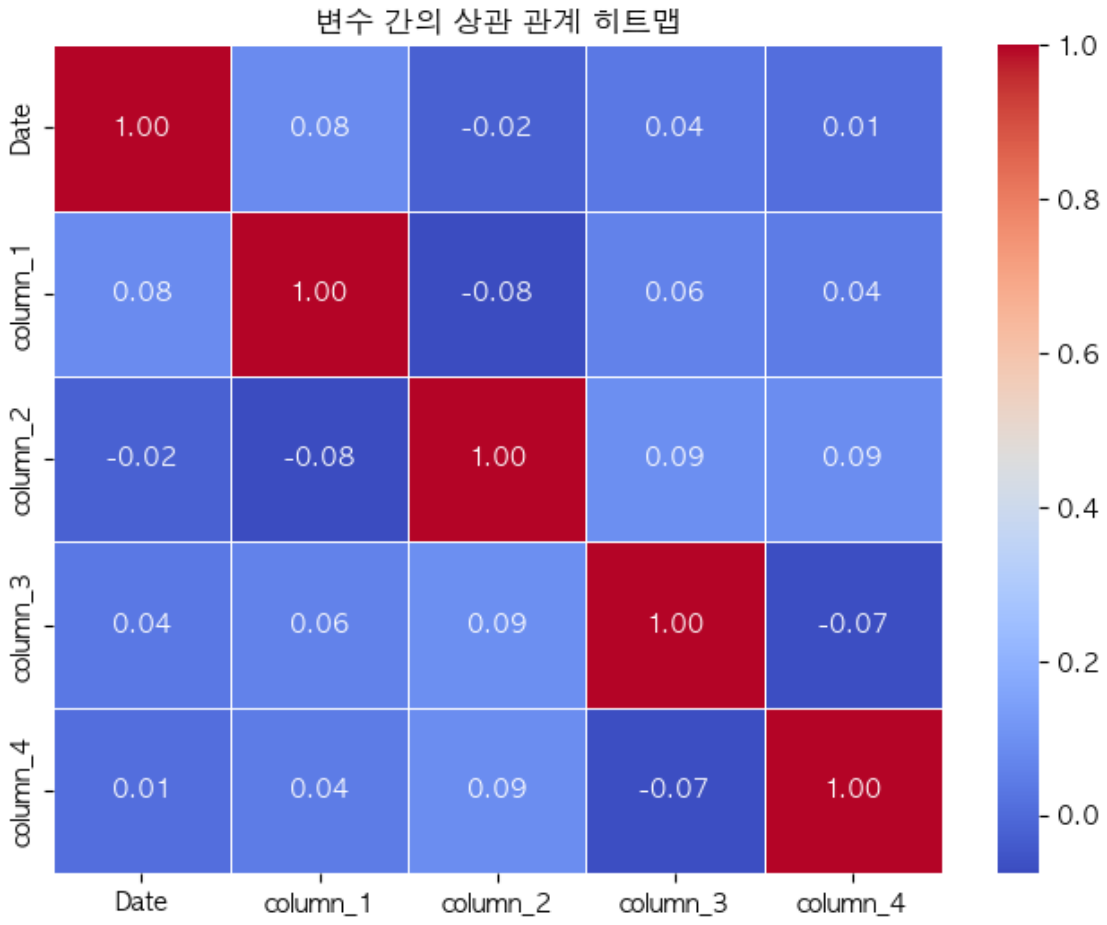
[4]:
```
#
#      -1  1        ,
# 1            ,
# -1
correlation_matrix = df.corr()  #
print("\n      :")
```

```
correlation_matrix
```

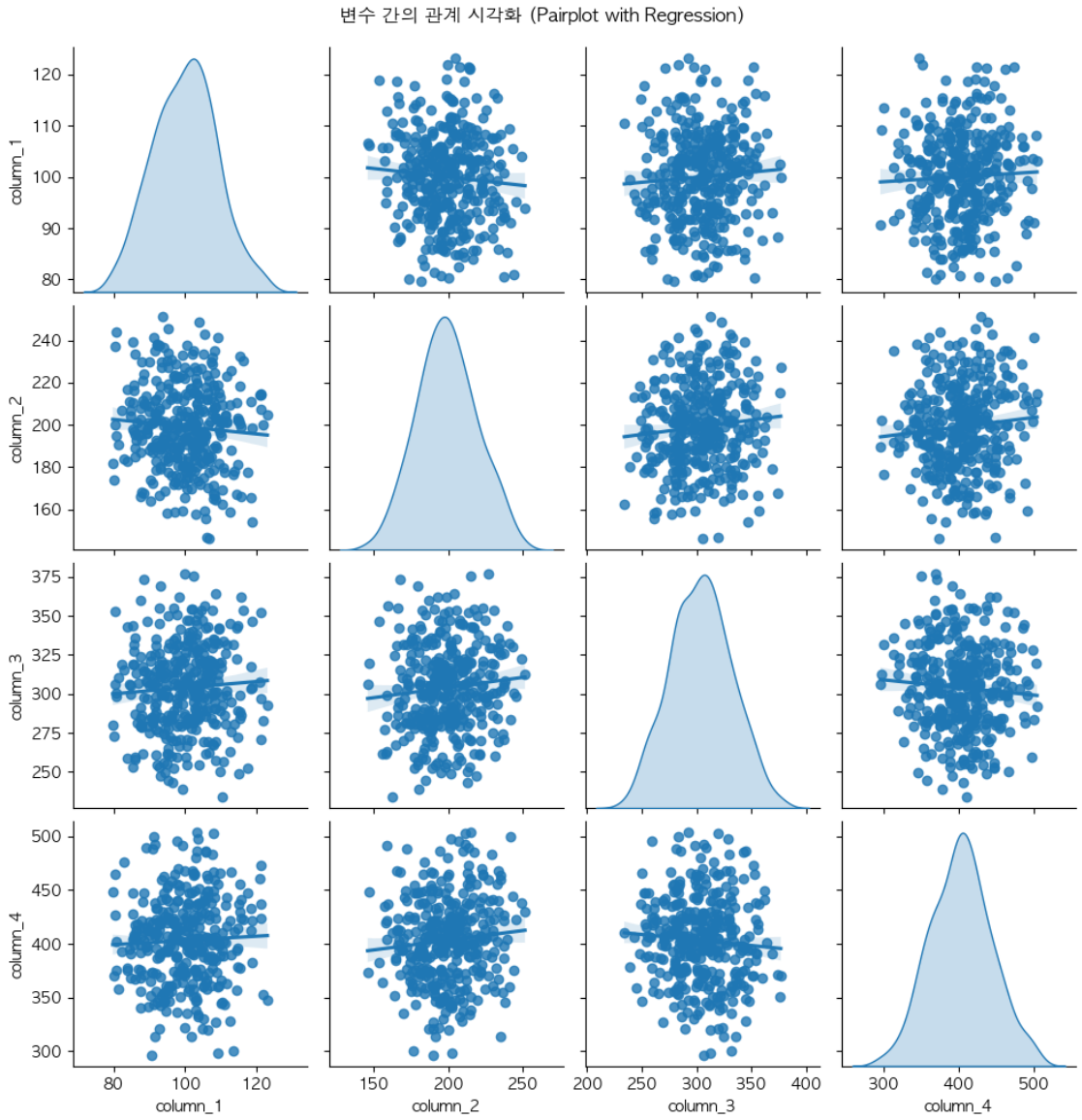
:

[4]:
```
              Date  column_1  column_2  column_3  column_4
Date      1.000000  0.084783 -0.023588  0.037085  0.009480
column_1  0.084783  1.000000 -0.075391  0.062603  0.042967
column_2 -0.023588 -0.075391  1.000000  0.094200  0.090470
column_3  0.037085  0.062603  0.094200  1.000000 -0.071123
column_4  0.009480  0.042967  0.090470 -0.071123  1.000000
```

[5]:
```
#
plt.figure(figsize=(8, 6))  #
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f',␣
 ↪linewidths=0.5)
plt.title("          ")
plt.show()
```



[7]:
```
#                (pairplot)
# =============    =============
#
#                    ,
# kind='reg'
# diag_kind='kde'
sns.pairplot(df.drop(columns=['Date']), kind='reg', diag_kind='kde')
plt.suptitle("         (Pairplot with Regression)", y=1.02)  #
plt.show()
```



[ ]: