الجمهورية اليمنية

جامعة إب

كلية العلوم

قسم علوم الحاسوب وتقنية المعلومات

تكليف مقرر

تنقيب بيانات - عملي

Data Mining

المحاضرة الثالثة

عمل الطالب :

أسامة سعيد محمد حمود سعيد - مجموعة A

إشراف :

أ. مالك المصنف

2024 - 2025

شرح خطوات ال Data Cleaning :

1 - Include Libraries :

استدعاء مكتبتي Numpy && Pandas لقراءة ملفات CSV من اجل عملية تنظيف البيانات

2 - Read Data From CSV file :

قراءة البيانات من داخل ملف CSV لمصفوفة للبدء بتنفيذ العمليات

3 - Know my Dataset :

التعرف على طبيعة البيانات التي سيتم العمل عليها من خلال معرفة الابعاد ، طباعة بعض القيم بداية ونهاية ال Data

4 - Get Columns Name :

معرفة أسماء الاعمدة للمصفوفة التي لدينا .

هذه الخطوات من اجل التعرف على طبيعة البيانات .

5 - Get the Dataset Information :

طباعة معلومات الاعمدة لمعرفة نوع البيانات و عدد الصفوف داخل كل عمود وكذلك حجم ال Data ك كل في الذاكرة

5 - Choose random of rows :

اختيار بيانات عشوائية لاجل معرفة طبيعة البيانات و بعض ال noise داخلها

6 - Get the Dataset Describe :

وصف البيانات من حيث العدد وتمركز البيانات والوسط واكبر واصغر قيمة لكل عمود

7 - Select Columns From Dataset with DataType :

اختيار بعض الاعمدة حسب نوع البيانات التي يتم اختيارها

8 - Get Unique Vlaue from Columns :

طباعة جميع القيم التي تحتويها كل عمود بحيث يتم طباعة القيم بدون تكرار

9 - Check if Customer_ID is not unique :

التحقق من ان عمود ال ID قيم غير مكررة – لكن تبين اننا لن نحتاجها عند استخلاص البيانات في النهاية

10 - Rename The Columns Name :

تغيير أسماء بعض الاعمدة وذلك من اجل ان تتماشى مع كتابة الكود البرمجي

11 - Copy my Dataset :

نسخ ال Dataset ل مصفوفة أخرى وذلك من اجل القيام بالعمليات مع الاحتفاظ بالنسخة الاصلية

12 - If I Use Drop Row for NaN :

نسخ ال dataset وعمل حذف للصفوف التي تحتوي على قيم فراغة وهذه العملية ليست صالحة لانها اضاعت نصف ال داتا

13 - CustomerID Columns :

معالجة عمود CustomerID حيث تم إزالة لقيم السالبة للحفول التي تحتوي عليها

13 - Gender Columns :

معالجة عمود Gender حيث تم جعل القيم كلها اما 0 للقيم female او انثى او 0 والقيمة 1 للقيم male و ذكر و 1

تحويل نوع البيانات للعمود الى int

14 - Age Columns :

معالجة عمود Age حيث تم إزالة القيم السالبة للحقول التي تحتوي عليها ، كذلك عمل المنوال للقيم الفارغة

15 - Annual_Income Columns :

معالجة عمود Annual Income حيث تم عمل قيمة الوسيط للعمود للقيم الفراغة ، و إزالة القيم السالبة للحقول التي تحتوي عليها

16 - Spending_Score Columns :

معالجة عمود Spending Score حيث تم عمل قيمة الوسيط للعمود للقيم الفراغة ، و إزالة القيم السالبة للحقول التي تحتوي عليها ، وتصحيح قيمة الحقول التي تحتوي على قيمة اكبر من المدى 1-100

17 - Dataset After Cleaning :

إعادة أسماء الاعمدة الاصلية وعرض ال Dataset بعد عملية المعالجة

17 - Write Dataset to CSV File :

حفظ ال Dataset المعالجة في ملف CSV

Include Libraries

```
import numpy as np
import pandas as pd
```

Read Data From CSV file

```
dataset =
pd.read_csv('Mall_Customers_toCleanAssignment.csv',encoding="ISO-8859-
1")
print (dataset)
```

```
     CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-
100)
0             1    Male  19.0                15.0
39.0
1             2       1  19.0                 NaN
39.0
2             3  Female  20.0                16.0
6.0
3             4    Male   NaN                15.0
39.0
4             5  Female   NaN                16.0
77.0
..          ...     ...   ...                 ...
...
995         996    ÃäËì  69.0                77.0
NaN
996         997    Male  50.0                 NaN
26.0
997         998    Male  67.0                44.0
69.0
998         999    Male   5.0               103.0
47.0
999        1000    ÐßÑ   25.0                38.0
82.0

[1000 rows x 5 columns]
```

Know my Dataset

```
 dataset.shape
```

```
(1000, 5)
```

```
 dataset.head()
```

```
   CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-
100)
```

```
0            1    Male  19.0                15.0
39.0
1            2       1  19.0                 NaN
39.0
2            3  Female  20.0                16.0
6.0
3            4    Male   NaN                15.0
39.0
4            5  Female   NaN                16.0
77.0
```

```
dataset.head(20)
```

```
    CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-
100)
0            1    Male  19.0                15.0
39.0
1            2       1  19.0                 NaN
39.0
2            3  Female  20.0                16.0
6.0
3            4    Male   NaN                15.0
39.0
4            5  Female   NaN                16.0
77.0
5            6  Female  22.0                17.0
76.0
6            7  Female   NaN                17.0
76.0
7            8  Female   NaN                16.0
177.0
8            9  Female  35.0               -18.0
NaN
9           10  Female  31.0                 NaN
40.0
10          11    Male  67.0                19.0
14.0
11          12  Female  35.0                19.0
99.0
12          13  Female  23.0                18.0
NaN
13          14  Female  30.0                 NaN
72.0
14          15  Female   NaN                16.0
6.0
15          16       1  22.0                20.0
79.0
16          17  Female -24.0                20.0
NaN
17          18    Male  20.0                21.0
```

```
66.0
18           19  Female   NaN                 17.0
40.0
19           20  Female   NaN                 17.0
76.0
```

```
 dataset.tail()
```

```
     CustomerID Gender   Age  Annual Income (k$)  Spending Score (1-
100)
995         996   ÃäËì  69.0                77.0
NaN
996         997   Male  50.0                 NaN
26.0
997         998   Male  67.0                44.0
69.0
998         999   Male   5.0               103.0
47.0
999        1000   ÐßÑ   25.0                38.0
82.0
```

```
dataset.tail(20)
```

```
     CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-
100)
980         981  Female  53.0                70.0
67.0
981         982  Female  23.0               136.0
76.0
982         983  Female   NaN                98.0
19.0
983         984    ÅäËì   8.0                 NaN
79.0
984         985  Female  61.0                51.0
42.0
985         986    Male  22.0                93.0
25.0
986         987  Female  50.0                95.0
64.0
987         988    Male   NaN                77.0
22.0
988         989  Female  11.0                30.0
16.0
989         990  Female  31.0                98.0
87.0
990         991  Female  40.0                31.0
NaN
991         992       0  11.0                75.0
NaN
992         993    Male   5.0               134.0
```

```
14.0
993         994  Female  65.0                64.0
30.0
994         995  Female   NaN                24.0
81.0
995         996    ÃäËì  69.0                77.0
NaN
996         997    Male  50.0                 NaN
26.0
997         998    Male  67.0                44.0
69.0
998         999    Male   5.0               103.0
47.0
999        1000    ÐßÑ   25.0                38.0
82.0
```

Get Columns Name

```
 dataset.columns
```

```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

Get the Dataset Information

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   CustomerID              1000 non-null   int64  
 1   Gender                  1000 non-null   object 
 2   Age                     851 non-null    float64
 3   Annual Income (k$)      844 non-null    float64
 4   Spending Score (1-100)  768 non-null    float64
dtypes: float64(3), int64(1), object(1)
memory usage: 39.2+ KB
```

Choose random of rows

```
dataset.sample(5)
```

```
     CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-
100)
159         160  Female  58.0                20.0
NaN
```

```
575          576    Male  57.0                106.0
NaN
394          395  Female  64.0                105.0
10.0
814          815    Male  69.0                 69.0
89.0
987          988    Male   NaN                 77.0
22.0
```

Get the Dataset Describe

```
 dataset.describe()
```

```
        CustomerID          Age  Annual Income (k$)  Spending Score (1-
100)
count  1000.000000  851.000000          844.000000
768.000000
mean    497.440000   37.301998           68.316351
49.990885
std     294.063632   18.470689           34.117906
28.787694
min    -652.000000  -53.000000          -43.000000               -
94.000000
25%     248.750000   23.000000           40.000000
27.000000
50%     498.500000   35.000000           65.000000
50.000000
75%     750.250000   53.500000           97.000000
73.000000
max    1000.000000   70.000000          137.000000
189.000000
```

Select Columns From Dataset with DataType

```
dataset.select_dtypes('int64').columns
```

```
Index(['CustomerID'], dtype='object')
```

Get Unique Vlaue from first Column

```
dataset.CustomerID.unique()
```

```
array([   1,    2,    3,    4,    5,    6,    7,    8,    9,   10,
11,
         12,   13,   14,   15,   16,   17,   18,   19,   20,   21,
22,
         23,   24,   25,   26,   27,   28,   29,   30,   31,   32,
33,
         34,   35,   36,   37,   38,   39,   40,   41,   42,   43,
```

```
44,
        45,   46,   47,   48,   49,   50,   51,   52,   53,   54,
55,
        56,   57,   58,   59,   60,   61,   62,   63,   64,   65,
66,
        67,   68,   69,   70,   71,   72,   73,   74,   75,   76,
77,
        78,   79,   80,   81,   82,   83,   84,   85,   86,   87,
88,
        89,   90,   91,   92,   93,   94,   95,   96,   97,   98,
99,
       100,  101,  102,  103,  104,  105,  106,  107,  108,  109,
110,
       111,  112,  113,  114,  115,  116,  117,  118,  119,  120,
121,
       122,  123,  124,  125,  126,  127,  128,  129,  130,  131,
132,
       133,  134,  135,  136,  137,  138,  139,  140,  141,  142,
143,
       144,  145,  146,  147,  148,  149,  150,  151,  152,  153,
154,
       155,  156,  157,  158,  159,  160,  161,  162,  163,  164,
165,
       166,  167,  168,  169,  170,  171,  172,  173,  174,  175,
176,
       177,  178,  179,  180,  181,  182,  183,  184,  185,  186,
187,
       188,  189,  190,  191,  192,  193,  194,  195,  196,  197,
198,
       199,  200,  201,  202,  203,  204,  205,  206,  207,  208,
209,
       210,  211,  212,  213,  214,  215,  216,  217,  218,  219,
220,
       221,  222,  223,  224,  225,  226,  227, -228,  229,  230,
231,
       232,  233,  234,  235,  236,  237,  238,  239,  240,  241,
242,
       243,  244,  245,  246,  247,  248,  249,  250,  251,  252,
253,
       254,  255,  256,  257,  258,  259,  260,  261,  262,  263,
264,
       265,  266,  267,  268,  269,  270,  271,  272,  273,  274,
275,
       276,  277,  278,  279,  280,  281,  282,  283,  284,  285,
286,
       287,  288,  289,  290,  291,  292,  293,  294,  295,  296,
297,
       298,  299,  300,  301,  302,  303,  304,  305,  306,  307,
308,
```

```
        309,  310,  311,  312,  313,  314,  315,  316,  317,  318,
319,
        320,  321,  322,  323,  324,  325,  326,  327,  328,  329,
330,
        331,  332,  333,  334,  335,  336,  337,  338,  339,  340,
341,
        342,  343,  344,  345,  346,  347,  348,  349,  350,  351,
352,
        353,  354,  355,  356,  357,  358,  359,  360,  361,  362,
363,
        364,  365,  366,  367,  368,  369,  370,  371,  372,  373,
374,
        375,  376,  377,  378,  379,  380,  381,  382,  383,  384,
385,
        386,  387,  388,  389,  390,  391,  392,  393,  394,  395,
396,
        397,  398,  399,  400,  401,  402,  403,  404,  405,  406,
407,
        408,  409,  410,  411,  412,  413,  414,  415,  416,  417,
418,
        419,  420,  421,  422,  423,  424,  425,  426,  427,  428,
429,
        430,  431,  432,  433,  434,  435,  436,  437,  438,  439,
440,
        441,  442,  443,  444,  445,  446,  447,  448,  449,  450,
451,
        452,  453,  454,  455,  456,  457,  458,  459,  460,  461,
462,
        463,  464,  465,  466,  467,  468,  469,  470,  471,  472,
473,
        474,  475,  476,  477,  478,  479,  480,  481,  482,  483,
484,
        485,  486,  487,  488,  489,  490,  491,  492,  493,  494,
495,
        496,  497,  498,  499,  500,  501,  502,  503,  504,  505,
506,
        507,  508,  509,  510,  511,  512,  513,  514,  515,  516,
517,
        518,  519,  520,  521,  522,  523,  524,  525,  526,  527,
528,
        529,  530,  531,  532,  533,  534,  535,  536,  537,  538,
539,
        540,  541,  542,  543,  544,  545,  546,  547,  548,  549,
550,
        551,  552,  553,  554,  555,  556,  557,  558,  559,  560,
561,
        562,  563,  564,  565,  566,  567,  568,  569,  570,  571,
572,
        573,  574,  575,  576,  577,  578,  579,  580,  581,  582,
```

```
583,
       584,  585,  586,  587,  588,  589,  590,  591,  592,  593,
594,
       595,  596,  597,  598,  599,  600,  601,  602,  603,  604,
605,
       606,  607,  608,  609,  610,  611,  612,  613,  614,  615,
616,
       617,  618,  619,  620,  621,  622,  623,  624,  625,  626,
627,
       628,  629,  630,  631,  632,  633,  634,  635,  636,  637,
638,
       639,  640,  641,  642,  643,  644,  645,  646,  647,  648,
649,
      -650,  651, -652,  653,  654,  655,  656,  657,  658,  659,
660,
       661,  662,  663,  664,  665,  666,  667,  668,  669,  670,
671,
       672,  673,  674,  675,  676,  677,  678,  679,  680,  681,
682,
       683,  684,  685,  686,  687,  688,  689,  690,  691,  692,
693,
       694,  695,  696,  697,  698,  699,  700,  701,  702,  703,
704,
       705,  706,  707,  708,  709,  710,  711,  712,  713,  714,
715,
       716,  717,  718,  719,  720,  721,  722,  723,  724,  725,
726,
       727,  728,  729,  730,  731,  732,  733,  734,  735,  736,
737,
       738,  739,  740,  741,  742,  743,  744,  745,  746,  747,
748,
       749,  750,  751,  752,  753,  754,  755,  756,  757,  758,
759,
       760,  761,  762,  763,  764,  765,  766,  767,  768,  769,
770,
       771,  772,  773,  774,  775,  776,  777,  778,  779,  780,
781,
       782,  783,  784,  785,  786,  787,  788,  789,  790,  791,
792,
       793,  794,  795,  796,  797,  798,  799,  800,  801,  802,
803,
       804,  805,  806,  807,  808,  809,  810,  811,  812,  813,
814,
       815,  816,  817,  818,  819,  820,  821,  822,  823,  824,
825,
       826,  827,  828,  829,  830,  831,  832,  833,  834,  835,
836,
       837,  838,  839,  840,  841,  842,  843,  844,  845,  846,
847,
```

```
        848,  849,  850,  851,  852,  853,  854,  855,  856,  857,
858,
        859,  860,  861,  862,  863,  864,  865,  866,  867,  868,
869,
        870,  871,  872,  873,  874,  875,  876,  877,  878,  879,
880,
        881,  882,  883,  884,  885,  886,  887,  888,  889,  890,
891,
        892,  893,  894,  895,  896,  897,  898,  899,  900,  901,
902,
        903,  904,  905,  906,  907,  908,  909,  910,  911,  912,
913,
        914,  915,  916,  917,  918,  919,  920,  921,  922,  923,
924,
        925,  926,  927,  928,  929,  930,  931,  932,  933,  934,
935,
        936,  937,  938,  939,  940,  941,  942,  943,  944,  945,
946,
        947,  948,  949,  950,  951,  952,  953,  954,  955,  956,
957,
        958,  959,  960,  961,  962,  963,  964,  965,  966,  967,
968,
        969,  970,  971,  972,  973,  974,  975,  976,  977,  978,
979,
        980,  981,  982,  983,  984,  985,  986,  987,  988,  989,
990,
        991,  992,  993,  994,  995,  996,  997,  998,  999, 1000],
      dtype=int64)
```

```
 dataset.CustomerID.value_counts()
```

```
CustomerID
1       1
672     1
659     1
660     1
661     1
       ..
339     1
340     1
341     1
342     1
1000    1
Name: count, Length: 1000, dtype: int64
```

Check if Customer_ID is not unique

```
if (dataset['CustomerID'].nunique() == len(dataset)):
    x = 1
x
```

```
1
```

```
 dataset.Gender.unique()
```

```
array(['Male', '1', 'Female', '0', 'ÐßÑ ', 'ÅäËì', 'ÃäËì'],
dtype=object)
```

```
 dataset.Age.unique()
```

```
array([ 19.,  20.,  nan,  22.,  35.,  31.,  67.,  23.,  30., -24.,
54.,
        53.,  58.,  21.,  42.,  36.,  40.,  48.,  49.,  50.,  27.,
59.,
        47.,  51., -53.,  70.,  63.,  18.,  33.,  68.,  60.,  46.,
52.,
        38.,  32.,  24.,  29.,  26.,  66.,  65., -36.,  55.,  28.,
25.,
        34.,  43.,  39.,  44.,  41.,  45.,  10.,  57.,   5.,   8.,
13.,
       -26.,  69.,  37.,  11.,   9.,  12.,   7.,  64.,   6.,  15.,
62.,
        16.,  61.,  14.,  56.,  17.])
```

◌ Rename The Columns Name

```
 dataset.rename(columns={'Annual Income (k$)':'Annual_Income'},
inplace=True)
 dataset.rename(columns={'Spending Score (1-100)':'Spending_Score'},
inplace=True)

dataset
```

```
     CustomerID  Gender   Age  Annual_Income  Spending_Score
0             1    Male  19.0           15.0            39.0
1             2       1  19.0            NaN            39.0
2             3  Female  20.0           16.0             6.0
3             4    Male   NaN           15.0            39.0
4             5  Female   NaN           16.0            77.0
..          ...     ...   ...            ...             ...
995         996    ÃäËì  69.0           77.0             NaN
996         997    Male  50.0            NaN            26.0
997         998    Male  67.0           44.0            69.0
998         999    Male   5.0          103.0            47.0
999        1000    ÐßÑ   25.0           38.0            82.0

[1000 rows x 5 columns]

dataset.Spending_Score.unique()

array([ 39.,   6.,  77.,  76., 177.,  nan,  40.,  14.,  99.,  72.,
79.,
```

```
        66.,  35.,  73.,  61.,   4.,  81.,  17.,  26.,  36.,  28.,
55.,
        47.,  94.,  52.,  54.,  60.,  41.,  50.,  51.,  46., 156.,
59.,
        48.,  42.,  49.,  56., -52.,  32., -94.,  43.,  57.,   5.,
11.,
        29.,  88.,  10.,  93.,  87.,  12.,  97.,  74.,  90.,  20.,
16.,
        89.,  83.,  27.,  75.,  13.,  86.,  92.,  15.,  68.,  85.,
18.,
       -51.,  24.,  30.,  38.,  22.,  23.,  62.,  69.,  45.,  53.,
34.,
        65.,   3.,  78.,  37.,  33.,  80.,  58.,   9.,  91.,   8.,
71.,
        44.,  21.,  82.,  84.,   7.,  19.,  63.,  67.,  31.,  25.,
95.,
        70.,  64.,  98., 189., 172., 135.])
```

```
dataset.Annual_Income.unique()
```

```
array([ 15.,  nan,  16.,  17., -18.,  19.,  18.,  20.,  21., -24.,
28.,
        25.,  24.,  33.,  34.,  37.,  29.,  39.,  40.,  42.,  43., -
43.,
        44.,  46.,  47.,  48.,  23.,  49.,  50.,  54.,  57.,  59.,
60.,
        62.,  63.,  64.,  65.,  67.,  70.,  71.,  73.,  74.,  75.,
76.,
        77.,  78.,  79.,  81.,  85.,  86.,  87.,  88.,  93.,  97.,
98.,
        99., 101., 103., 120., 126., 137., 116.,  96., 134., 128.,
52.,
        38.,  82.,  51.,  31., 127., 108., 110., 111.,  30.,  27.,
95.,
       113.,  53., 119.,  68., 102., 106.,  92., 130., 114.,  90.,
115.,
       100., 121.,  72., 117., 118., 107.,  89., 133., 109.,  45.,
69.,
        26., 105., 136., 124.,  55.,  66.,  58., 132., 112.,  22.,
131.,
        94.,  35.,  56.,  41., 129., 104.,  36., 123., 135.,  61.,
32.,
        83.,  84.,  80.,  91.])
```

```
 dataset.isnull().sum()
```

```
CustomerID          0
Gender              0
Age               149
Annual_Income     156
```

```
Spending_Score    232
dtype: int64
```

Copy my Dataset

```
dataset_copy = dataset.copy()

dataset_copy
```

```
     CustomerID  Gender   Age  Annual_Income  Spending_Score
0             1    Male  19.0           15.0            39.0
1             2       1  19.0            NaN            39.0
2             3  Female  20.0           16.0             6.0
3             4    Male   NaN           15.0            39.0
4             5  Female   NaN           16.0            77.0
..          ...     ...   ...            ...             ...
995         996    ÃäËì  69.0           77.0             NaN
996         997    Male  50.0            NaN            26.0
997         998    Male  67.0           44.0            69.0
998         999    Male   5.0          103.0            47.0
999        1000    ÐßÑ   25.0           38.0            82.0

[1000 rows x 5 columns]
```

If I Use Drop Row for NaN

```
df_cleaned = dataset_copy.dropna()

df_cleaned
```

```
     CustomerID  Gender   Age  Annual_Income  Spending_Score
0             1    Male  19.0           15.0            39.0
2             3  Female  20.0           16.0             6.0
5             6  Female  22.0           17.0            76.0
10           11    Male  67.0           19.0            14.0
11           12  Female  35.0           19.0            99.0
..          ...     ...   ...            ...             ...
992         993    Male   5.0          134.0            14.0
993         994  Female  65.0           64.0            30.0
997         998    Male  67.0           44.0            69.0
998         999    Male   5.0          103.0            47.0
999        1000    ÐßÑ   25.0           38.0            82.0

[463 rows x 5 columns]
```

```
df_cleaned.isnull().sum()
```

```
CustomerID        0
Gender            0
Age               0
```

```
Annual_Income     0
Spending_Score    0
dtype: int64
```

```
dataset_copy
```

	CustomerID	Gender	Age	Annual_Income	Spending_Score
0	1	Male	19.0	15.0	39.0
1	2	1	19.0	NaN	39.0
2	3	Female	20.0	16.0	6.0
3	4	Male	NaN	15.0	39.0
4	5	Female	NaN	16.0	77.0
..	...	...	...	...	...
995	996	ÃäËì	69.0	77.0	NaN
996	997	Male	50.0	NaN	26.0
997	998	Male	67.0	44.0	69.0
998	999	Male	5.0	103.0	47.0
999	1000	ÐßÑ	25.0	38.0	82.0

```
[1000 rows x 5 columns]
```

CustomerID Columns

Multiplie CustomerID *-1 If the Value <0 - Handling Wrong Format

```
for i in range (0,len(dataset_copy)):
    if dataset_copy.loc[i,'CustomerID'] < 0:
        dataset_copy.loc[i,'CustomerID']*=-1


print ('CustomerID before : \n')
print (dataset.CustomerID.unique(),'\n')

print ('CustomerID after  : \n')
dataset_copy.CustomerID.unique()
```

```
CustomerID before :
```

```
[   1    2    3    4    5    6    7    8    9   10   11   12   13   14
   15   16   17   18   19   20   21   22   23   24   25   26   27   28
   29   30   31   32   33   34   35   36   37   38   39   40   41   42
   43   44   45   46   47   48   49   50   51   52   53   54   55   56
   57   58   59   60   61   62   63   64   65   66   67   68   69   70
   71   72   73   74   75   76   77   78   79   80   81   82   83   84
   85   86   87   88   89   90   91   92   93   94   95   96   97   98
   99  100  101  102  103  104  105  106  107  108  109  110  111  112
  113  114  115  116  117  118  119  120  121  122  123  124  125  126
  127  128  129  130  131  132  133  134  135  136  137  138  139  140
  141  142  143  144  145  146  147  148  149  150  151  152  153  154
  155  156  157  158  159  160  161  162  163  164  165  166  167  168
```

169	170	171	172	173	174	175	176	177	178	179	180	181	182
183	184	185	186	187	188	189	190	191	192	193	194	195	196
197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224
225	226	227	-228	229	230	231	232	233	234	235	236	237	238
239	240	241	242	243	244	245	246	247	248	249	250	251	252
253	254	255	256	257	258	259	260	261	262	263	264	265	266
267	268	269	270	271	272	273	274	275	276	277	278	279	280
281	282	283	284	285	286	287	288	289	290	291	292	293	294
295	296	297	298	299	300	301	302	303	304	305	306	307	308
309	310	311	312	313	314	315	316	317	318	319	320	321	322
323	324	325	326	327	328	329	330	331	332	333	334	335	336
337	338	339	340	341	342	343	344	345	346	347	348	349	350
351	352	353	354	355	356	357	358	359	360	361	362	363	364
365	366	367	368	369	370	371	372	373	374	375	376	377	378
379	380	381	382	383	384	385	386	387	388	389	390	391	392
393	394	395	396	397	398	399	400	401	402	403	404	405	406
407	408	409	410	411	412	413	414	415	416	417	418	419	420
421	422	423	424	425	426	427	428	429	430	431	432	433	434
435	436	437	438	439	440	441	442	443	444	445	446	447	448
449	450	451	452	453	454	455	456	457	458	459	460	461	462
463	464	465	466	467	468	469	470	471	472	473	474	475	476
477	478	479	480	481	482	483	484	485	486	487	488	489	490
491	492	493	494	495	496	497	498	499	500	501	502	503	504
505	506	507	508	509	510	511	512	513	514	515	516	517	518
519	520	521	522	523	524	525	526	527	528	529	530	531	532
533	534	535	536	537	538	539	540	541	542	543	544	545	546
547	548	549	550	551	552	553	554	555	556	557	558	559	560
561	562	563	564	565	566	567	568	569	570	571	572	573	574
575	576	577	578	579	580	581	582	583	584	585	586	587	588
589	590	591	592	593	594	595	596	597	598	599	600	601	602
603	604	605	606	607	608	609	610	611	612	613	614	615	616
617	618	619	620	621	622	623	624	625	626	627	628	629	630
631	632	633	634	635	636	637	638	639	640	641	642	643	644
645	646	647	648	649	-650	651	-652	653	654	655	656	657	658
659	660	661	662	663	664	665	666	667	668	669	670	671	672
673	674	675	676	677	678	679	680	681	682	683	684	685	686
687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714
715	716	717	718	719	720	721	722	723	724	725	726	727	728
729	730	731	732	733	734	735	736	737	738	739	740	741	742
743	744	745	746	747	748	749	750	751	752	753	754	755	756
757	758	759	760	761	762	763	764	765	766	767	768	769	770
771	772	773	774	775	776	777	778	779	780	781	782	783	784
785	786	787	788	789	790	791	792	793	794	795	796	797	798
799	800	801	802	803	804	805	806	807	808	809	810	811	812
813	814	815	816	817	818	819	820	821	822	823	824	825	826
827	828	829	830	831	832	833	834	835	836	837	838	839	840
841	842	843	844	845	846	847	848	849	850	851	852	853	854

```
  855  856  857  858  859  860  861  862  863  864  865  866  867  868
  869  870  871  872  873  874  875  876  877  878  879  880  881  882
  883  884  885  886  887  888  889  890  891  892  893  894  895  896
  897  898  899  900  901  902  903  904  905  906  907  908  909  910
  911  912  913  914  915  916  917  918  919  920  921  922  923  924
  925  926  927  928  929  930  931  932  933  934  935  936  937  938
  939  940  941  942  943  944  945  946  947  948  949  950  951  952
  953  954  955  956  957  958  959  960  961  962  963  964  965  966
  967  968  969  970  971  972  973  974  975  976  977  978  979  980
  981  982  983  984  985  986  987  988  989  990  991  992  993  994
  995  996  997  998  999 1000]

CustomerID after  :


array([   1,    2,    3,    4,    5,    6,    7,    8,    9,   10,
11,
         12,   13,   14,   15,   16,   17,   18,   19,   20,   21,
22,
         23,   24,   25,   26,   27,   28,   29,   30,   31,   32,
33,
         34,   35,   36,   37,   38,   39,   40,   41,   42,   43,
44,
         45,   46,   47,   48,   49,   50,   51,   52,   53,   54,
55,
         56,   57,   58,   59,   60,   61,   62,   63,   64,   65,
66,
         67,   68,   69,   70,   71,   72,   73,   74,   75,   76,
77,
         78,   79,   80,   81,   82,   83,   84,   85,   86,   87,
88,
         89,   90,   91,   92,   93,   94,   95,   96,   97,   98,
99,
        100,  101,  102,  103,  104,  105,  106,  107,  108,  109,
110,
        111,  112,  113,  114,  115,  116,  117,  118,  119,  120,
121,
        122,  123,  124,  125,  126,  127,  128,  129,  130,  131,
132,
        133,  134,  135,  136,  137,  138,  139,  140,  141,  142,
143,
        144,  145,  146,  147,  148,  149,  150,  151,  152,  153,
154,
        155,  156,  157,  158,  159,  160,  161,  162,  163,  164,
165,
        166,  167,  168,  169,  170,  171,  172,  173,  174,  175,
176,
        177,  178,  179,  180,  181,  182,  183,  184,  185,  186,
187,
        188,  189,  190,  191,  192,  193,  194,  195,  196,  197,
```

```
198,
      199,  200,  201,  202,  203,  204,  205,  206,  207,  208,
209,
      210,  211,  212,  213,  214,  215,  216,  217,  218,  219,
220,
      221,  222,  223,  224,  225,  226,  227,  228,  229,  230,
231,
      232,  233,  234,  235,  236,  237,  238,  239,  240,  241,
242,
      243,  244,  245,  246,  247,  248,  249,  250,  251,  252,
253,
      254,  255,  256,  257,  258,  259,  260,  261,  262,  263,
264,
      265,  266,  267,  268,  269,  270,  271,  272,  273,  274,
275,
      276,  277,  278,  279,  280,  281,  282,  283,  284,  285,
286,
      287,  288,  289,  290,  291,  292,  293,  294,  295,  296,
297,
      298,  299,  300,  301,  302,  303,  304,  305,  306,  307,
308,
      309,  310,  311,  312,  313,  314,  315,  316,  317,  318,
319,
      320,  321,  322,  323,  324,  325,  326,  327,  328,  329,
330,
      331,  332,  333,  334,  335,  336,  337,  338,  339,  340,
341,
      342,  343,  344,  345,  346,  347,  348,  349,  350,  351,
352,
      353,  354,  355,  356,  357,  358,  359,  360,  361,  362,
363,
      364,  365,  366,  367,  368,  369,  370,  371,  372,  373,
374,
      375,  376,  377,  378,  379,  380,  381,  382,  383,  384,
385,
      386,  387,  388,  389,  390,  391,  392,  393,  394,  395,
396,
      397,  398,  399,  400,  401,  402,  403,  404,  405,  406,
407,
      408,  409,  410,  411,  412,  413,  414,  415,  416,  417,
418,
      419,  420,  421,  422,  423,  424,  425,  426,  427,  428,
429,
      430,  431,  432,  433,  434,  435,  436,  437,  438,  439,
440,
      441,  442,  443,  444,  445,  446,  447,  448,  449,  450,
451,
      452,  453,  454,  455,  456,  457,  458,  459,  460,  461,
462,
```

```
        463,  464,  465,  466,  467,  468,  469,  470,  471,  472,
473,
        474,  475,  476,  477,  478,  479,  480,  481,  482,  483,
484,
        485,  486,  487,  488,  489,  490,  491,  492,  493,  494,
495,
        496,  497,  498,  499,  500,  501,  502,  503,  504,  505,
506,
        507,  508,  509,  510,  511,  512,  513,  514,  515,  516,
517,
        518,  519,  520,  521,  522,  523,  524,  525,  526,  527,
528,
        529,  530,  531,  532,  533,  534,  535,  536,  537,  538,
539,
        540,  541,  542,  543,  544,  545,  546,  547,  548,  549,
550,
        551,  552,  553,  554,  555,  556,  557,  558,  559,  560,
561,
        562,  563,  564,  565,  566,  567,  568,  569,  570,  571,
572,
        573,  574,  575,  576,  577,  578,  579,  580,  581,  582,
583,
        584,  585,  586,  587,  588,  589,  590,  591,  592,  593,
594,
        595,  596,  597,  598,  599,  600,  601,  602,  603,  604,
605,
        606,  607,  608,  609,  610,  611,  612,  613,  614,  615,
616,
        617,  618,  619,  620,  621,  622,  623,  624,  625,  626,
627,
        628,  629,  630,  631,  632,  633,  634,  635,  636,  637,
638,
        639,  640,  641,  642,  643,  644,  645,  646,  647,  648,
649,
        650,  651,  652,  653,  654,  655,  656,  657,  658,  659,
660,
        661,  662,  663,  664,  665,  666,  667,  668,  669,  670,
671,
        672,  673,  674,  675,  676,  677,  678,  679,  680,  681,
682,
        683,  684,  685,  686,  687,  688,  689,  690,  691,  692,
693,
        694,  695,  696,  697,  698,  699,  700,  701,  702,  703,
704,
        705,  706,  707,  708,  709,  710,  711,  712,  713,  714,
715,
        716,  717,  718,  719,  720,  721,  722,  723,  724,  725,
726,
        727,  728,  729,  730,  731,  732,  733,  734,  735,  736,
```

```
737,
       738,  739,  740,  741,  742,  743,  744,  745,  746,  747,
748,
       749,  750,  751,  752,  753,  754,  755,  756,  757,  758,
759,
       760,  761,  762,  763,  764,  765,  766,  767,  768,  769,
770,
       771,  772,  773,  774,  775,  776,  777,  778,  779,  780,
781,
       782,  783,  784,  785,  786,  787,  788,  789,  790,  791,
792,
       793,  794,  795,  796,  797,  798,  799,  800,  801,  802,
803,
       804,  805,  806,  807,  808,  809,  810,  811,  812,  813,
814,
       815,  816,  817,  818,  819,  820,  821,  822,  823,  824,
825,
       826,  827,  828,  829,  830,  831,  832,  833,  834,  835,
836,
       837,  838,  839,  840,  841,  842,  843,  844,  845,  846,
847,
       848,  849,  850,  851,  852,  853,  854,  855,  856,  857,
858,
       859,  860,  861,  862,  863,  864,  865,  866,  867,  868,
869,
       870,  871,  872,  873,  874,  875,  876,  877,  878,  879,
880,
       881,  882,  883,  884,  885,  886,  887,  888,  889,  890,
891,
       892,  893,  894,  895,  896,  897,  898,  899,  900,  901,
902,
       903,  904,  905,  906,  907,  908,  909,  910,  911,  912,
913,
       914,  915,  916,  917,  918,  919,  920,  921,  922,  923,
924,
       925,  926,  927,  928,  929,  930,  931,  932,  933,  934,
935,
       936,  937,  938,  939,  940,  941,  942,  943,  944,  945,
946,
       947,  948,  949,  950,  951,  952,  953,  954,  955,  956,
957,
       958,  959,  960,  961,  962,  963,  964,  965,  966,  967,
968,
       969,  970,  971,  972,  973,  974,  975,  976,  977,  978,
979,
       980,  981,  982,  983,  984,  985,  986,  987,  988,  989,
990,
       991,  992,  993,  994,  995,  996,  997,  998,  999, 1000],
      dtype=int64)
```

Gender Column

Set Values for Range 0 , 1

```
for i in range (0,len(dataset_copy)):
    if dataset_copy.loc[i,'Gender']== 'Male' or
dataset_copy.loc[i,'Gender']== 'ÅäËì' or
dataset_copy.loc[i,'Gender']=='ÃäËì' or
dataset_copy.loc[i,'Gender']=='1' :
        dataset_copy.loc[i,'Gender'] = 1
    elif dataset_copy.loc[i,'Gender']== 'Female' or
dataset_copy.loc[i,'Gender']== 'ÐßÑ ' or
dataset_copy.loc[i,'Gender']=='0':
                dataset_copy.loc[i,'Gender'] = 0

print ('Gender before : \n')
print (dataset.Gender.unique(),'\n')

print ('Gender after  : \n')
dataset_copy.Gender.unique()
```

```
Gender before :

['Male' '1' 'Female' '0' 'ÐßÑ ' 'ÅäËì' 'ÃäËì']

Gender after  :


array([1, 0], dtype=object)
```

Set DataType For Column

```
dataset_copy['Gender']=dataset_copy['Gender'].astype(int)

print(dataset_copy.Gender.unique())
print (dataset_copy.Gender.dtypes)
```

```
[1 0]
int32
```

Age Column

Multiple the Value *-1 if Value < 0

```
for i in range (0,len(dataset_copy)):
    if dataset_copy.loc[i,'Age'] < 0:
        dataset_copy.loc[i,'Age']*=-1

print ('Age before : \n')
print (dataset.Age.unique(),'\n')
```

```
print ('Age after  : \n')
dataset_copy.Age.unique()
```

```
Age before :

[ 19.  20.  nan  22.  35.  31.  67.  23.  30. -24.  54.  53.  58.  21.
  42.  36.  40.  48.  49.  50.  27.  59.  47.  51. -53.  70.  63.  18.
  33.  68.  60.  46.  52.  38.  32.  24.  29.  26.  66.  65. -36.  55.
  28.  25.  34.  43.  39.  44.  41.  45.  10.  57.   5.   8.  13. -26.
  69.  37.  11.   9.  12.   7.  64.   6.  15.  62.  16.  61.  14.  56.
  17.]

Age after  :


array([19., 20., nan, 22., 35., 31., 67., 23., 30., 24., 54., 53.,
58.,
       21., 42., 36., 40., 48., 49., 50., 27., 59., 47., 51., 70.,
63.,
       18., 33., 68., 60., 46., 52., 38., 32., 29., 26., 66., 65.,
55.,
       28., 25., 34., 43., 39., 44., 41., 45., 10., 57.,  5.,  8.,
13.,
       69., 37., 11.,  9., 12.,  7., 64.,  6., 15., 62., 16., 61.,
14.,
       56., 17.])
```

Put the Mode Insted of NaN Value

```
dataset_copy['Age']=dataset_copy['Age'].fillna(value =
dataset_copy['Age'].mode()[0])

dataset_copy.Age.unique()
```

```
array([19., 20., 67., 22., 35., 31., 23., 30., 24., 54., 53., 58.,
21.,
       42., 36., 40., 48., 49., 50., 27., 59., 47., 51., 70., 63.,
18.,
       33., 68., 60., 46., 52., 38., 32., 29., 26., 66., 65., 55.,
28.,
       25., 34., 43., 39., 44., 41., 45., 10., 57.,  5.,  8., 13.,
69.,
       37., 11.,  9., 12.,  7., 64.,  6., 15., 62., 16., 61., 14.,
56.,
       17.])
```

Annual_Income Column

Put the Median instead of NaN Values

```
dataset_copy['Annual_Income']=dataset_copy['Annual_Income'].fillna(val
ue = dataset_copy['Annual_Income'].median())

 dataset_copy.isnull().sum()
```

```
CustomerID          0
Gender              0
Age                 0
Annual_Income       0
Spending_Score    232
dtype: int64
```

Multiple the Value *-1 if Value < 0

```
for i in range (0,len(dataset_copy)):
    if dataset_copy.loc[i,'Annual_Income'] < 0:
        dataset_copy.loc[i,'Annual_Income']*=-1

print ('Annual_Income before : \n')
print (dataset.Annual_Income.unique(),'\n')

print ('Annual_Income after  : \n')
dataset_copy.Annual_Income.unique()
```

```
Annual_Income before :

[ 15.  nan  16.  17. -18.  19.  18.  20.  21. -24.  28.  25.  24.  33.
  34.  37.  29.  39.  40.  42.  43. -43.  44.  46.  47.  48.  23.  49.
  50.  54.  57.  59.  60.  62.  63.  64.  65.  67.  70.  71.  73.  74.
  75.  76.  77.  78.  79.  81.  85.  86.  87.  88.  93.  97.  98.  99.
 101. 103. 120. 126. 137. 116.  96. 134. 128.  52.  38.  82.  51.  31.
 127. 108. 110. 111.  30.  27.  95. 113.  53. 119.  68. 102. 106.  92.
 130. 114.  90. 115. 100. 121.  72. 117. 118. 107.  89. 133. 109.  45.
  69.  26. 105. 136. 124.  55.  66.  58. 132. 112.  22. 131.  94.  35.
  56.  41. 129. 104.  36. 123. 135.  61.  32.  83.  84.  80.  91.]

Annual_Income after  :


array([ 15.,  65.,  16.,  17.,  18.,  19.,  20.,  21.,  24.,  28.,
25.,
        33.,  34.,  37.,  29.,  39.,  40.,  42.,  43.,  44.,  46.,
47.,
        48.,  23.,  49.,  50.,  54.,  57.,  59.,  60.,  62.,  63.,
64.,
        67.,  70.,  71.,  73.,  74.,  75.,  76.,  77.,  78.,  79.,
81.,
```

```
        85.,  86.,  87.,  88.,  93.,  97.,  98.,  99., 101., 103.,
120.,
       126., 137., 116.,  96., 134., 128.,  52.,  38.,  82.,  51.,
31.,
       127., 108., 110., 111.,  30.,  27.,  95., 113.,  53., 119.,
68.,
       102., 106.,  92., 130., 114.,  90., 115., 100., 121.,  72.,
117.,
       118., 107.,  89., 133., 109.,  45.,  69.,  26., 105., 136.,
124.,
        55.,  66.,  58., 132., 112.,  22., 131.,  94.,  35.,  56.,
41.,
       129., 104.,  36., 123., 135.,  61.,  32.,  83.,  84.,  80.,
91.])
```

Spending_Score Column

Put the Median instead of NaN Values

```
dataset_copy['Spending_Score']=dataset_copy['Spending_Score'].fillna(v
alue = dataset_copy['Spending_Score'].median())

 dataset_copy.isnull().sum()
```

```
CustomerID        0
Gender            0
Age               0
Annual_Income     0
Spending_Score    0
dtype: int64
```

Multiple the Value *-1 if Value < 0

```
for i in range (0,len(dataset_copy)):
    if dataset_copy.loc[i,'Spending_Score'] < 0:
        dataset_copy.loc[i,'Spending_Score']*=-1

print ('Spending_Score before : \n')
print (dataset.Spending_Score.unique(),'\n')

print ('Spending_Score after  : \n')
dataset_copy.Spending_Score.unique()
```

```
Spending_Score before :

[ 39.   6.  77.  76. 177.  nan  40.  14.  99.  72.  79.  66.  35.  73.
  61.   4.  81.  17.  26.  36.  28.  55.  47.  94.  52.  54.  60.  41.
  50.  51.  46. 156.  59.  48.  42.  49.  56. -52.  32. -94.  43.  57.
   5.  11.  29.  88.  10.  93.  87.  12.  97.  74.  90.  20.  16.  89.
  83.  27.  75.  13.  86.  92.  15.  68.  85.  18. -51.  24.  30.  38.
```

```
  22.  23.  62.  69.  45.  53.  34.  65.   3.  78.  37.  33.  80.  58.
   9.  91.   8.  71.  44.  21.  82.  84.   7.  19.  63.  67.  31.  25.
  95.  70.  64.  98. 189. 172. 135.]

Spending_Score after  :


array([ 39.,   6.,  77.,  76., 177.,  50.,  40.,  14.,  99.,  72.,
79.,
        66.,  35.,  73.,  61.,   4.,  81.,  17.,  26.,  36.,  28.,
55.,
        47.,  94.,  52.,  54.,  60.,  41.,  51.,  46., 156.,  59.,
48.,
        42.,  49.,  56.,  32.,  43.,  57.,   5.,  11.,  29.,  88.,
10.,
        93.,  87.,  12.,  97.,  74.,  90.,  20.,  16.,  89.,  83.,
27.,
        75.,  13.,  86.,  92.,  15.,  68.,  85.,  18.,  24.,  30.,
38.,
        22.,  23.,  62.,  69.,  45.,  53.,  34.,  65.,   3.,  78.,
37.,
        33.,  80.,  58.,   9.,  91.,   8.,  71.,  44.,  21.,  82.,
84.,
         7.,  19.,  63.,  67.,  31.,  25.,  95.,  70.,  64.,  98.,
189.,
       172., 135.])
```

Make the Values -100 if Value > 100

```
for i in range (0,len(dataset_copy)):
    if dataset_copy.loc[i,'Spending_Score'] > 100:
        dataset_copy.loc[i,'Spending_Score']-=100

print ('Spending_Score before : \n')
print (dataset.Spending_Score.unique(),'\n')

print ('Spending_Score after  : \n')
dataset_copy.Spending_Score.unique()
```

```
Spending_Score before :

[ 39.   6.  77.  76. 177.  nan  40.  14.  99.  72.  79.  66.  35.  73.
  61.   4.  81.  17.  26.  36.  28.  55.  47.  94.  52.  54.  60.  41.
  50.  51.  46. 156.  59.  48.  42.  49.  56. -52.  32. -94.  43.  57.
   5.  11.  29.  88.  10.  93.  87.  12.  97.  74.  90.  20.  16.  89.
  83.  27.  75.  13.  86.  92.  15.  68.  85.  18. -51.  24.  30.  38.
  22.  23.  62.  69.  45.  53.  34.  65.   3.  78.  37.  33.  80.  58.
   9.  91.   8.  71.  44.  21.  82.  84.   7.  19.  63.  67.  31.  25.
  95.  70.  64.  98. 189. 172. 135.]
```

```
Spending_Score after  :


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

```
 dataset_copy.isnull().sum()
```

```
CustomerID        0
Gender            0
Age               0
Annual_Income     0
Spending_Score    0
dtype: int64
```

Dataset After Cleaning

Return the Column Name

```
 dataset_copy.rename(columns={'Annual_Income':'Annual Income (k$)'},
inplace=True)
 dataset_copy.rename(columns={'Spending_Score':'Spending Score (1-
100)'}, inplace=True)

dataset_copy
```

```
     CustomerID  Gender   Age  Annual Income (k$)  Spending Score (1-
100)
0             1       1  19.0                15.0
39.0
1             2       1  19.0                65.0
39.0
2             3       0  20.0                16.0
6.0
3             4       1  67.0                15.0
39.0
4             5       0  67.0                16.0
```

```
77.0
..           ...     ...   ...                  ...
...
995          996       1  69.0                 77.0
50.0
996          997       1  50.0                 65.0
26.0
997          998       1  67.0                 44.0
69.0
998          999       1   5.0                103.0
47.0
999         1000       0  25.0                 38.0
82.0

[1000 rows x 5 columns]
```

Write Dataset to CSV File

```
dataset_copy.to_csv('datanew2.csv')

print ('save Succesfully')
```

```
save Succesfully
```