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

Random sampling is one of the easiest from of collecting data from the total population, under random samploing each

Number of the population carries an equal opportunity of being chosen as a part of the sampling process'''

In [2]:
```
x = pd.read_csv('C:/Users/ADMIN/OneDrive/Desktop/Employee_monthly_salary.csv')
print(x.head())
print(x.shape)
print(x.sample(200))
print(x)
```

```
   EmpID            Name Gender Date_of_Birth  Age   Join_Date  \
0  19575    Keven Norman      M    03-09-1994   25  02-12-2019
1  19944  Kristin Werner      F    23-06-1994   26  13-01-2020
2  20055    Avery Barber      M    27-02-1996   24  14-11-2019
3  20058    Boris Gibson      M    29-09-1993   26  13-01-2020
4  20332       Leif Mack      M    01-05-1991   29  04-06-2018

   Tenure_in_org_in_months   GROSS  Net_Pay  Deduction  Deduction_percentage
\
0                        7   74922    71494       3428                  4.58
1                        6   44375    39971       4404                  9.92
2                        8   82263    77705       4558                  5.54
3                        6   44375    40164       4211                  9.49
4                       25  235405   143963      91442                 38.84

                           Designation                    Department
0    Product Operations Analyst.Associate.    IT Product Management & Ops
1  Platform Operations Engineer.Associate.           Platform Operations
2  Platform Operations Engineer.Associate.           Platform Operations
3  Platform Operations Engineer.Associate.           Platform Operations
4               Software Engineer.Senior.  Enterprise Access Engineering
(1802, 13)
      EmpID               Name Gender Date_of_Birth  Age   Join_Date  \
343    8328        Jamey Leach      M    05-02-1980   40  28-03-2013
327   22449  Jerry Fitzpatrick      F    21-04-1992   28  09-09-2019
218   11024       Elvis Gaines      M    27-02-1989   31  25-08-2014
545    8911       Israel Wyatt      M    22-11-1989   30  18-07-2013
922    6400    Antoine Cordova      M    21-06-1985   35  29-08-2011
...     ...                ...    ...           ...  ...         ...
1352  20814  Ricardo Zimmerman      M    21-10-1991   28  10-09-2018
1220  19086       Pablo Bowers      M    18-02-1994   26  28-09-2017
400   21419        Otha Rhodes      M    10-12-1989   30  24-01-2019
1529  22013      Efrain Terrell      M    16-07-1987   32  13-06-2019
528   14630     Cameron Durham      M    28-02-1984   36  13-07-2015

      Tenure_in_org_in_months   GROSS  Net_Pay  Deduction  \
343                        87  211594    95281     116313
327                        10  142833   128202      14631
218                        70  209503   134803      74700
545                        83  155130   125345      29785
922                       106  285798   121640     164158
...                       ...     ...      ...        ...
1352                       22   99846    83647      16199
1220                       33   81323    72189       9134
400                        17  155524   130436      25088
1529                       13  230273   140034      90239
528                        60  173960   112098      61862

      Deduction_percentage                        Designation  \
343                  54.97              Supervisor..Help Desk
```

```
327                10.24  Business Operations Analyst.Senior.
218                35.66            Software Engineer.Senior.
545                19.20   Hardware Performance Engineer II..
922                57.44                 Manager..Engineering
...                  ...                                  ...
1352               16.22          Media Operations Engineer..
1220               11.23       Network Engagement Consultant..
400                16.13        Technical Project Manager II..
1529               39.19         Program Manager II..Technical
528                35.56  Business Operations Analyst.Senior.

                              Department
343     Enterprise Infrastructure Services
327                 Marketing - Operations
218                       Corporate Systems
545                  Networks - Technology
922                       Corporate Systems
...                                    ...
1352                      Amatec - BOCC/EMM
1220          Networks - APJ Infrastructure
400                     Web Americas - ECG
1529  Media Engineering Program Management
528                Media Division Sales Ops

[200 rows x 13 columns]
      EmpID              Name Gender Date_of_Birth  Age   Join_Date  \
0     19575      Keven Norman      M    03-09-1994   25  02-12-2019
1     19944    Kristin Werner      F    23-06-1994   26  13-01-2020
2     20055      Avery Barber      M    27-02-1996   24  14-11-2019
3     20058      Boris Gibson      M    29-09-1993   26  13-01-2020
4     20332         Leif Mack      M    01-05-1991   29  04-06-2018
...     ...               ...    ...           ...  ...         ...
1797  18835  Darius Wilkerson      M    14-01-1991   29  21-08-2017
1798  19066     Erick Ballard      M    29-08-1992   27  25-09-2017
1799  21644   Lawerence Downs      M    05-07-1991   29  01-04-2019
1800  19673     Abdul Watkins      M    19-08-1972   47  26-12-2017
1801  19790   Chase Fernandez      M    20-03-1993   27  22-01-2018

      Tenure_in_org_in_months   GROSS  Net_Pay  Deduction  \
0                           7   74922    71494       3428
1                           6   44375    39971       4404
2                           8   82263    77705       4558
3                           6   44375    40164       4211
4                          25  235405   143963      91442
...                       ...     ...      ...        ...
1797                       34   88934    88734        200
1798                       33  133224   133024        200
1799                       15   72547    71246       1301
1800                       30  227176   220778       6398
1801                       29  114641   114441        200

      Deduction_percentage                          Designation  \
0                     4.58    Product Operations Analyst.Associate.
1                     9.92  Platform Operations Engineer.Associate.
2                     5.54  Platform Operations Engineer.Associate.
3                     9.49  Platform Operations Engineer.Associate.
```

```
4                     38.84            Software Engineer.Senior.
...                     ...                                    ...
1797                   0.22       Technical Solutions Engineer..
1798                   0.15               Software Engineer II..
1799                   1.79         Business Operations Analyst..
1800                   2.82           Manager..Account Management
1801                   0.17                      Order Analyst..

                                    Department
0                  IT Product Management & Ops
1                          Platform Operations
2                          Platform Operations
3                          Platform Operations
4                Enterprise Access Engineering
...                                        ...
1797                         AmaTec - EMEA TSE
1798             GSS EPIC Engineering (HC COGS)
1799                    Marketing - Operations
1800                             Americas- AMG
1801  Finance - Customer Revenue Operations G&A

[1802 rows x 13 columns]
```

systematic sampling

systematic sampling is a probablity samploing method where elements from a target population are chosen -->

by selecting a random.starting point and selecting sample members after a fixed sampling interval'''

we dpo systematic sampling, choosing every 10th element.

In [3]:
```
print(x.iloc[0:1802:10])
```

```
      EmpID             Name Gender Date_of_Birth  Age   Join_Date  \
0     19575     Keven Norman      M    03-09-1994   25  02-12-2019
10    22612         Ola Lara      F    01-11-1992   27  26-09-2019
20    22750      Long Forbes      M    02-01-1993   27  28-10-2019
30    22788    Herman Hester      M    09-09-1982   37  04-11-2019
40    22816    Damian Molina      M    23-08-1990   29  11-11-2019
...     ...              ...    ...           ...  ...         ...
1760  21370  Dewey Stephens      M    23-09-1998   21  08-07-2019
1770  21383    Tobias Hurley      M    16-07-1998   21  08-07-2019
1780  22359      Colby Hines      M    25-08-1994   25  19-08-2019
1790  15349    Trevor Tanner      M    20-05-1984   36  23-11-2015
1800  19673    Abdul Watkins      M    19-08-1972   47  26-12-2017

      Tenure_in_org_in_months   GROSS  Net_Pay  Deduction  \
0                           7   74922    71494       3428
10                          9   99552    88551      11001
20                          8  199333   139639      59694
30                          8  175533   140203      35330
40                          8  215200   167585      47615
...                       ...     ...      ...        ...
1760                       12   73813    57813      16000
1770                       12   70813    54940      15873
1780                       10  138867   101954      36913
1790                       55   26796    24325       2471
1800                       30  227176   220778       6398

      Deduction_percentage                                Designation
\
0                     4.58          Product Operations Analyst.Associate.
10                   11.05                 Technical Solutions Engineer..
20                   29.95                        Data Scientist.Senior.
30                   20.13            Business Operations Analyst.Senior.
40                   22.13                     Software Engineer.Senior.
...                    ...                                            ...
1760                 21.68                            Software Engineer..
1770                 22.42                            Software Engineer..
1780                 26.58                               Data Scientist..
1790                  9.22  Software Development Engineer in Test.Senior II.
1800                  2.82                 Manager..Account Management

                      Department
0     IT Product Management & Ops
10              AmaTec - EMEA TSE
20           Security Engineering
30          Web Division Sales Ops
40           Security Engineering
...                           ...
1760      Enterprise Applications
1770                AmaTec - APS
1780         Security Engineering
1790            Enterprise Center
1800                Americas- AMG

[181 rows x 13 columns]
```

Stratified sampling

In [4]:
```
x_males=x[x['Gender']=='M']
x_males
x_males.sample(100)

x_females=x[x['Gender']=='F']
x_females
x_females.sample(100)
x_females.shape[1]#column wise
x_females.shape[0]# row wise
```

Out[4]: 499

how can we get cluster samples:

create an array in form of dictionary

In [6]:
```
import numpy as np
```

In [7]:
```
x={'N_numbers':np.arange(1,16)}
print(x)
```

```
{'N_numbers': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14,
15])}
```

In [8]:
```
x={'employee_id':np.arange(1,21),'value':np.random.randn(20)}
y=pd.DataFrame(x)
x
y.sample(20)
y
```

Out[8]:

	employee_id	value
0	1	0.519258
1	2	-0.528575
2	3	1.826504
3	4	-0.578494
4	5	-0.172233
5	6	0.361477
6	7	-0.934163
7	8	-1.618655
8	9	0.519454
9	10	-0.079292
10	11	-1.024189
11	12	0.794662
12	13	1.669983
13	14	-1.489024
14	15	-0.646080
15	16	-1.670078
16	17	0.722429
17	18	-0.482364
18	19	-0.830799
19	20	-0.656316

In [6]:
```
import numpy as np
```

In [7]:
```
x=[32,111,138,28,59,77,97]
y = np.var(x)
print(y)
```

```
1432.2448979591834
```

In [10]:
```
x=[32,111,138,28,59,77,97]
```

In [11]:
```
y=np.std(x)
```

In [12]:
```
print(y)
```

```
37.84501153334721
```

Find the mean in this data set

In [13]:
```
x=[23,45,66,77,88,100,23]
mean=sum(x)/7
print(mean)
```

```
60.285714285714285
```

find the median

In [15]:
```
x=[23,45,66,77,88,100,23]
```

In [16]:
```
print(np.median(x))
```

```
66.0
```

find the mode

In [35]:
```
import pandas as pd
```

In [38]:
```
x=pd.Series([23,45,66,77,88,100,23])
print(x.mode())
```

```
0    23
dtype: int64
```

find the standard

In [24]:
```
x=[23,45,66,77,88,100,23]
```

In [25]:
```
y=np.std(x)
```

In [26]:
```
print(y)
```

```
28.464084064530393
```

find the variance

In [27]:
```
x=[23,45,66,77,88,100,23]
```

In [28]:
```
y=np.var(x)
```

In [29]:
```
print(y)
```

```
810.2040816326531
```

RANGE

In [42]:
```
# How can we calculate range?
# To calulate range we will take the highest number from the set and then subs
# number from the set.

# set_1=66,67,67,68,68,68,69,69,69
# set_2=70,70,71,71,72,73,75

# find the minimun value=66

# find the maximum value=75

# find the difference(75-66)=9

# 9 is the range
```

In [41]:
```

# Range:

# The range in statistics for a given data set is the difference between the h
# for example, if the given data set is {2,5,8,10,3}, then the range will be 1
```

Interquartile range

In [43]:
```
# The IQR is used to measure how spread out
```

In [44]:
```
from scipy import stats
```

In [45]:
```
x=[32,36,46,47,56,69,75,79,79,88,89,91,92,93,96,97,101,105,112,116]
```

In [46]:
```
IQR=stats.iqr(x,interpolation='midpoint')
```

In [47]:
```
print(IQR)
```

```
34.0
```

find the central measurement of this data set

In [1]:
```
x=[12,45,67,89,12,34,56,23,12,12,89,56]
mean=sum(x)/12
print(mean)
```

```
42.25
```

In [4]:
```
import numpy as np
```

In [5]:
```
x=[12,45,67,89,12,34,56,23,12,12,89,56]
print(np.median(x))
```

```
39.5
```

In [6]:
```
import pandas as pd
```

In [8]:
```
x=pd.Series([12,45,67,89,12,34,56,23,12,12,89,56])
print(x.mode())
```

```
0    12
dtype: int64
```

find the spread measurement of this data set

In [14]:
```
x=[34,22,12,22,33,44,55,66,12]
y=np.std(x)
print(y)
```

```
17.69494591998266
```

In [15]:
```
x=[34,22,12,22,33,44,55,66,12]
y=np.var(x)
print(y)
```

```
313.1111111111111
```

In [12]:
```
from scipy import stats
```

In [16]:
```
x=[34,22,12,22,33,44,55,66,12]
IQR=stats.iqr(x,interpolation='midpoint')
print(IQR)
```

```
22.0
```

find the sampling

In [39]:
```
x=pd.read_csv("E:/DATA SCIENCE/october/diabetes.csv")
print(x.head())
print(x.shape)
print(x.sample(200))
print(x)
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
(768, 9)
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
421            2       94             68             18       76  26.0
157            1      109             56             21      135  25.2
40             3      180             64             25       70  34.0
365            5       99             54             28       83  34.0
429            1       95             82             25      180  35.0
..           ...      ...            ...            ...      ...   ...
150            1      136             74             50      204  37.4
542           10       90             85             32        0  34.9
46             1      146             56              0        0  29.7
480            3      158             70             30      328  35.5
297            0      126             84             29      215  30.7

     DiabetesPedigreeFunction  Age  Outcome
421                     0.561   21        0
157                     0.833   23        0
40                      0.271   26        0
365                     0.499   30        0
429                     0.233   43        1
..                        ...  ...      ...
150                     0.399   24        0
542                     0.825   56        1
46                      0.564   29        0
480                     0.344   35        1
297                     0.520   24        0

[200 rows x 9 columns]
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
```

```
765            5      121             72            23      112  26.2
766            1      126             60             0        0  30.1
767            1       93             70            31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]
```

systematic sampling

In [40]:
```
print(x.iloc[0:755:10])
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
10             4      110             92              0        0  37.6
20             3      126             88             41      235  39.3
30             5      109             75             26        0  36.0
40             3      180             64             25       70  34.0
..           ...      ...            ...            ...      ...   ...
710            3      158             64             13      387  31.2
720            4       83             86             19        0  29.3
730            3      130             78             23       79  28.4
740           11      120             80             37      150  42.3
750            4      136             70              0        0  31.2

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
10                      0.191   30        0
20                      0.704   27        0
30                      0.546   60        0
40                      0.271   26        0
..                        ...  ...      ...
710                     0.295   24        0
720                     0.317   34        0
730                     0.323   34        1
740                     0.785   48        1
750                     1.182   22        1

[76 rows x 9 columns]
```

cluster sampling

In [41]:
```
x={'N_numbers':np.arange(1,756)}
print(x)
```

```
{'N_numbers': 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])}
```

In [48]:

```
x={'Pregnancies':np.arange(1,21),'value':np.random.randn(20)}
y=pd.DataFrame(x)
print(y)
print(y.sample(20))
```

```
    Pregnancies     value
0             1 -1.053300
1             2 -1.223002
2             3  0.185040
3             4  0.680077
4             5 -0.400433
5             6 -0.506382
6             7 -0.093845
7             8  0.877761
8             9  0.253763
9            10 -1.032179
10           11 -0.210180
11           12 -0.897414
12           13 -0.292399
13           14  0.227485
14           15  1.083488
15           16  0.165697
16           17  0.578198
17           18  0.534838
18           19 -0.824654
19           20  0.529509
    Pregnancies     value
11           12 -0.897414
19           20  0.529509
13           14  0.227485
2             3  0.185040
0             1 -1.053300
6             7 -0.093845
18           19 -0.824654
14           15  1.083488
5             6 -0.506382
1             2 -1.223002
16           17  0.578198
4             5 -0.400433
3             4  0.680077
9            10 -1.032179
12           13 -0.292399
8             9  0.253763
7             8  0.877761
15           16  0.165697
10           11 -0.210180
17           18  0.534838
```

stratified sampling

In [52]:
```
x=pd.read_csv("E:/DATA SCIENCE/october/diabetes.csv")
outcomes=x[x['Outcome']==1]
print(outcomes)
out=outcomes.sample(67)
print(out)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
2              8      183             64              0        0  23.3
4              0      137             40             35      168  43.1
6              3       78             50             32       88  31.0
8              2      197             70             45      543  30.5
..           ...      ...            ...            ...      ...   ...
755            1      128             88             39      110  36.5
757            0      123             72              0        0  36.3
759            6      190             92              0        0  35.5
761            9      170             74             31        0  44.0
766            1      126             60              0        0  30.1

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
2                       0.672   32        1
4                       2.288   33        1
6                       0.248   26        1
8                       0.158   53        1
..                        ...  ...      ...
755                     1.057   37        1
757                     0.258   52        1
759                     0.278   66        1
761                     0.403   43        1
766                     0.349   47        1

[268 rows x 9 columns]
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
266            0      138              0              0        0  36.3
4              0      137             40             35      168  43.1
17             7      107             74              0        0  29.6
404            5      168             64              0        0  32.9
493            4      125             70             18      122  28.9
..           ...      ...            ...            ...      ...   ...
236            7      181             84             21      192  35.9
701            6      125             78             31        0  27.6
276            7      106             60             24        0  26.5
100            1      163             72              0        0  39.0
22             7      196             90              0        0  39.8

     DiabetesPedigreeFunction  Age  Outcome
266                     0.933   25        1
4                       2.288   33        1
17                      0.254   31        1
404                     0.135   41        1
493                     1.144   45        1
..                        ...  ...      ...
236                     0.586   51        1
701                     0.565   49        1
```

```
276                    0.296   29       1
100                    1.222   33       1
22                     0.451   41       1

[67 rows x 9 columns]
```

Normal distribution:

In [1]:
```
# Normal distribution also known as the gaussian distribution,is a probability
# about the mean, showing that data near the mean are more frequent in occurre
# in graphical from the normal distribution appears as a "bell curve"
```

What is the difference between percentage and percentile

In [2]:
```
# If you calculate the percentage of a single person or individual base this i
# If you calculate the total number of stundent 95% percentage then, that is c
```

What is the meaning of empirical rule?

In [3]:
```
# The empirical rule, or the 68-95-97.7 rule, tells you where the most of the
```

What is the meaning of standard deviation

In [4]:
```
# Standard deviation is a statistic that measure the dispersion of a dataset r
```

What is percentile?

In [5]:
```
# You got the percentage of total numbers persons
# In percentage you get individual percentage.You got 95 percentile means you
# In statistic, a percentile is a term that describe how a score compares to o
```

In [9]:
```
import numpy as np
```

In [13]:
```
x=[43,45,45,50,50,53,58,66,69,73,75,77,78,81,87,89,92,94,94,97]
z=np.percentile(x,75)
print(z,"75th percentile of x")
```

```
87.5 75th percentile of x
```

In [16]:
```
x=[75,77,78,78,80,81,81,82,83,84,84,84,85,87,87,88,88,88,89,90]
z=np.percentile(x,20)
print(z,"20th percentile of x")
```

```
79.6 20th percentile of x
```

Percentile can be calculated using the formula

In [15]:
```
# n=(p/100) x N,
# where p=percentile, N=number of values in a data set
```

In [20]:
```
from scipy import stats
import numpy as np
```

In [33]:
```
data=np.array([6,7,7,12,13,13,15,16,19,22])
a=stats.zscore(data)
b=np.std(data)
c=np.mean(data)
print(a)
print(b)
print(c)
```

```
[-1.39443338 -1.19522861 -1.19522861 -0.19920477  0.          0.
  0.39840954  0.5976143   1.19522861  1.79284291]
5.019960159204453
13.0
```

In [1]:
```
# what is the z score,
# with the help of z score we can find the area of curve in form of standard d
# z score can be positive and negative.

# formula to find the z score

# z=x-mean/std.
```

In [2]:
```
# What do u mean by hypothesis testing?
# Hypothesis testing is a from of statistical inference that uses data from a
# about a population parameter or a population probablity distribution.
```

In [4]:
```
# Alternate hypothesis: There is two variable sun and tree this is my hypothes
# between sun and tree, and tree are dependent variable and sun is independent
# NULL hypothesis: Now the null hypothesis said there is no relation between s
# null hypothesis said there is no connection between two or more variables.
```

In [5]:
```
# WHY DO WE NEED NULL HYPOTHESIS?
# Null hypothesis said there is no realtionship between two variables, for res
# we select this thought that there is no relation between two variables,thats
# we denote null hypothesis as h0 and alternate hypothesis as h1.
```

In [6]:
```
# WHAT IS P VALUE?
# P value is probability of NULL hypothesis being True.
# wioth the help of p value either you accept the null hypothesis or reject th

# If u have a significant value level and data then there is two ways, accept 

# null hypothesis denote h0
# alternate hypotheesis denote h1

# the answer will be accept/reject

# accept: said u accept the null hypotheisis whatever it says
# reject: said u accept the alternate hypothesis.


# What is p value?

# Now i have to do some test on this data like T test,chi-sqaure, anova, z tes
# we can obtained the p value.

# What is p value? It is the probablity thaat you will obatain a test result g
```

In [7]:
```
# How can you signify null hypotheisis will accept or reject
# Typical significance levels are:

# 0.1(10%)
# 0.05(5%)
# 0.01(1%)

# if p value <=0.01 then you have very strong case against null hypothesis

# if this range will be 0.01 <=p value <=0.05 means there is strong evidence a

# if there is range will be 0.05 <= p value <=0.1 means there is mild evidence

# if there is p value >=10 and more than 0.1 then means there is no evidence a
#  can accept the null hypothesis.
```

In [8]:
```
# What is a T-test in python?
# The indepentent t-test is a parameter test used to test for a statistically 
# between 2 groups.

# What is t-test?

# A t-test is an inferential statistic used to determine if there is a signifi
# two groups and how they are related.
```

In [18]:
```
import scipy.stats as stats
import numpy as np
```

In [23]:
```
x=np.array([14,15,15,16,13,8,14,17,16,14,19,20,21,15,15,16,16,13,14,12])
y=np.array([15,17,14,17,8,12,19,19,14,17,22,24,16,13,16,13,18,15,13,16])

print(np.var(x),np.var(y))
z=stats.ttest_ind(a=x, b=y, equal_var=True)
print(z)
```

```
7.727500000000001 12.09
Ttest_indResult(statistic=-0.7343678075051265, pvalue=0.46723220591335846)
```

In [24]:
```
# Interpreting the result:

# This is the time to analyze the result. The p-value of the test comes out to
# than the significant level alpha(that is, 0.05).This implies that we can say
# one class is statistically not different from the average height of student


# Here, since the p-value (0.53004) is greater than alpha =0.05 so we cannot r

# we do not have sufficient evidence to say that the mean height of students b
```

In [28]:
```
from scipy.stats import f_oneway
```

In [30]:
```
# the very first step is to create three arrays that will keep the information
# performance when each of the engine
# oil is applied

performance1=[89,89,88,78,79]
performance2=[93,92,94,89,88]
performance3=[89,88,89,93,90]
performance4=[81,78,81,92,82]

# step 2:conduct the one waay ANOVA:
# python provides us f_oneway() funtion from scipy library using which we can
```

In [32]:
```
# conduct one-way anova
f_oneway(performance1,performance2,performance3,performance4)


# ANALYZE THE RESULT:

# The statistic and p-value turn out to be equal to 4.625 and 0.016336459 resp
# hence we would reject the null hypothesis.This implies that we have sufficen
# in the performance among four diffrernt engine oils.
```

Out[32]:
```
F_onewayResult(statistic=4.625000000000002, pvalue=0.016336459839780215)
```

In [1]:
```
# What is chi-square in data science?
# A chi-square test is a statistical test used to compare observed result with
# is tyo determine if a difference between expected results.

# What is chi-sqaure in ml?
# A chi-square is used in standard to etst the indepenence of two events.Given
# and expected counte E.
```

In [2]:
```
# HO = There is no link betwwen gender and political party preference
# H1 = There is a link between gender and political party preference.
```

In [3]:
```
# calculate the expected value:(row total)*(column total)/total numbers of obs
```

In [4]:
```
# 200*240/440=109
# 130*200/440=59
# 50*200/440=22.72
# 240*220/440=120
# 130*240/440=65
# 50*220/440=25
```

In [5]:
```
# What is mann-whitney u test?
# The Mann-Whitney U Test, also known as the Wilcoxon Rank Sum Test, is a non-


# The Mann-Whitney U Test assesses whether two sampled groups are likely to de
#     The null hypothesis (H0) is that the two populations are equal.
#     The alternative hypothesis (H1) is that the two populations are not equa


# Some researchers interpret this as comparing the medians between the two pop

# When to use the Mann-Whitney U Test

# Non-parametric tests (sometimes referred to as ‘distribution-free tests’) ar
```

In [6]:
```
# What is mann-whitney u test?
# The mann-whitney u test is a non-parameteric test that can be used in place
# the null hypothesis that two samples come from the same population.

# Mann-whitney u test is used for every field,but is frequently used in psycho
# other disciplines. fro exmaple in psychology it isused to compare attitude o
# the effect of medicines.
```

In [7]:
```
import scipy.stats as stats
```

In [8]:
```
group1=[20,23,21,25,18,17,18,24,20,24,23,19]
group2=[24,25,21,22,23,18,17,28,24,27,21,23]
print(stats.mannwhitneyu(group1,group2,alternative='two-sided'))
```

```
MannwhitneyuResult(statistic=50.0, pvalue=0.21138945901258455)
```

In [9]:
```
# The test statistic is 50.0 and the corresponding two-sided p-value is 0.2114
# since the p-value (0.2114) is not less then 0.05, we fail to reject the null
# THis mean we do not have sufficent evidence to say that the true mean mgp is
```

In [10]:
```
# what is kruskal-wallis test?
# a researcher wants to know whether or not three drug havew different effects
# who all experience similar knee pain and randomnly splits them up into three
# drug 2, drug 3.
```

In [14]:
```
import scipy.stats as stats
```

In [15]:
```
data_group1=[7,9,12,15,21]
data_group2=[5,8,14,13,25]
data_group3=[6,8,8,9,5]
print(stats.kruskal(data_group1,data_group2,data_group3))
```

```
KruskalResult(statistic=3.492418772563175, pvalue=0.17443390338074047)
```

In [16]:
```
x=[7,9,12,15,21]
y=[5,8,14,13,25]
z=[6,8,8,9,5]
g=stats.kruskal(x,y,x)
print(g)
```

```
KruskalResult(statistic=0.015135135135135707, pvalue=0.9924609943783124)
```

In [1]:
```
from scipy.stats import chisquare
```

In [2]:
```
a=chisquare([16,18,16,14,12,12])
print(a)
```

```
Power_divergenceResult(statistic=2.0, pvalue=0.8491450360846096)
```

Definition of F-test:

In [4]:
```
# In statistics, a test statistic has an F -distribution under the null hypoth
# It is used to compare the statistical models as per the data set available.


# Formula for f-test to compare two variables.
# A statistical F test uses an F statistic to compare two variance, σ1 and σ2,
# The result will always be a postive number because varianves are alwyas post
# thus the equation for comparing two variables with the f-test.


# f=s²1/s²2
```

In [6]:
```
from scipy import stats
import numpy as np
```

In [7]:
```
x=[7,9,12,15,21]
y=[5,8,14,13,25]
z=np.array(x)
r=np.array(y)
f=np.var(z)/np.var(r)
print(f)
```

```
0.5162393162393164
```

classwork

In [8]:
```
# find the mean
x=[77,78,85,86,86,86,87,87,88,94,99,103]
mean=sum(x)/12
print(mean)
```

```
88.0
```

In [10]:
```
# find the mode
import pandas as pd
x=pd.Series([77,78,85,86,86,86,87,87,88,94,99,103])
print(x.mode())
```

```
0    86
dtype: int64
```

In [13]:
```
# find the median
import numpy as np
x=[77,78,85,86,86,86,87,87,88,94,99,103]
print(np.median(x))
```

```
86.5
```

In [15]:
```
# find the standard
x=[77,78,85,86,86,86,87,87,88,94,99,103]
y=np.std(x)
print(y)
```

```
7.22264956006817
```

In [16]:
```
# find the variance
x=[77,78,85,86,86,86,87,87,88,94,99,103]
y=np.var(x)
print(y)
```

```
52.166666666666664
```

In [17]:
```
# find the IQR
x=[77,78,85,86,86,86,87,87,88,94,99,103]
IQR=stats.iqr(x,interpolation='midpoint')
print(IQR)
```

```
5.5
```

In [18]:
```
# find the range max
x=[77,78,85,86,86,86,87,87,88,94,99,103]
y=np.max(x)
print(y)
```

```
103
```

In [19]:
```
# find the range min
x=[77,78,85,86,86,86,87,87,88,94,99,103]
y=np.min(x)
print(y)
```

```
77
```

Parameter and non-parameter

In [20]:
```
# specific assumption are made about the population parameter

# ratio and interval scale
# require more information for calculation
# assume a regular bell-shaped curve distribution

# more statistical power

# less robust

# result can be generalised


# NON-parametric:

# No assumption are made about the population parameter.

# Nominal and ordinal scale.

# require less information for calculation.

# do not assume a regular bell shaped curve of distribution.

# less powerfull

# more robust

# result can be generalized.
```

In [21]:
```
# Regressiom analysis is a statistical method to model the relationship betwee
# variables with one or more independent variables.specifically, regression an
# the dependent variable is changing corresponding to an independent variable 
# fixed. Its predicts continous/real values such as temperature,age,salary,pri

# Example: suppose there is a marketing company A, who does variable advertise
# the advertisement made by the company in the last 5 years and the correspond

# Types of regression:

# Linear regression in machine learning: linear regression is one of the easie
# it is a statistical method that is used for predicitive analysis.linear regr
# numeric variables such as sales,salary,age,product price ,etc.

# Linear regression algorrithm shows a linear relatiuonship between a depenten
# hence called as linear regression.

# types of linear regression

# linear regression can be further divided into two types of the algorithm:

# simple linear regression:
# If a single independent variable is  used to predict the value of a numeric 

# multiple linear regression:

# if more than one independent varibale is used to predict the value of a nume
# a linear regression algorithm is called multiple linear regression.


# what is logistic regression?

# Logistic regression is one of the most popular machine learning algorithm,wh
# it is used for predicting the categorial dependent variable using a given se

# logistic regression is much similar to the linear regression except that how
# linear regression is used for solving regression problems,whereas logistic r
# problems.

# for example , a logistic regression could be used to predict whethere a poli
# or whether a high school student will be admitted or not to a particular col
# decision between two alternatives.
```

In [ ]: