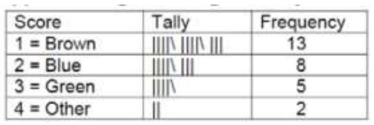
(1) Given the following setup {Class, Tally score, Frequency}, develop an application that generates the table shown; (you can populate the relevant data; minimum data size :50 records). The table is only an illustration for a data of color scores, you are free to test the application over any data set with the application generating the tally and frequency scores.



Score	Tally	Frequency
1 = Brown	||||\ ||||\ |||	13
2 = Blue	||||\ |||	8
3 = Green	||||\	5
4 = Other	||	2

```
import pandas as pd



DataCSV=pd.read_csv('colors.csv',na_values=["NA"]) #blank cells read as NAN
FreqTable=(pd.DataFrame({'Frequency' : DataCSV.groupby( [ "Colour"] ).size()}).reset_index())


tallies=[]
for count in FreqTable["Frequency"]:
    tally=""
    for i in range(1,count+1):
        if(i%5==0 ):
            tally=tally+'\ '
        else:
            tally=tally+'|'
    tallies.append((tally))


FreqTable["Tallies"]=tallies
print(FreqTable)
```

```
          Colour  Frequency     Tallies
0     Aquamarine          9  ||||\ ||||
1           Blue          6      ||||\ |
2        Crimson          1           |
3         Fuscia          2          ||
4      Goldenrod          2          ||
5          Green          3         |||
6         Indigo          5       ||||\
7          Khaki          2          ||
8         Maroon          1           |
9           Mauv          5       ||||\
10        Orange          6      ||||\ |
11          Pink          2          ||
12          Puce          3         |||
13           Red          4        ||||
14          Teal          7     ||||\ ||
15     Turquoise          9  ||||\ ||||
16        Violet          8    ||||\ |||
17        Yellow          4        ||||
```

(2) In a class of 18 students, assume marks distribution in an exam are as follows. Let the roll numbers start with CSE20D01 and all the odd roll numbers secure marks as follows: 25+((i+7)%10) and even roll numbers : 25+((i+8)%10). Develop an application that sets up the data and calculate the mean and median for the marks obtained using the platform support.

```
import numpy as np

marks = {}
roll="CSE20D0"
num=1
for i in range(19):
    if num%2==0:
        marks[roll+str(num)] = 25+((num+8)%10)
    else:
        marks[roll+str(num)] = 25+((num+7)%10)
    print(roll+str(num), marks[roll+str(num)])
    num+=1

mark=list((marks.values()))
print("Mean:", np.mean(mark), "Median:",np.median(mark))
```

```
CSE20D01 33
CSE20D02 25
```

```
CSE20D03 25
CSE20D04 27
CSE20D05 27
CSE20D06 29
CSE20D07 29
CSE20D08 31
CSE20D09 31
CSE20D010 33
CSE20D011 33
CSE20D012 25
CSE20D013 25
CSE20D014 27
CSE20D015 27
CSE20D016 29
CSE20D017 29
CSE20D018 31
CSE20D019 31
Mean: 28.789473684210527 Median: 29.0
```

(3) For a sample space of 20 elements, the values are fitted to the line Y=2X+3, X>5. Develop an application that sets up the data and computes the standard deviation of this sample space. (use random number generator supported in your development platform to generate values of X).

```
import random
import statistics

x = []
y = []
for i in range(20):
    x.append(random.randint(6,100))
    y.append((2*x[i])+3)

stddev = statistics.stdev(y)
print(x)
print(y)
print("Standard Deviation",stddev)
```

```
[76, 61, 56, 61, 83, 46, 53, 46, 79, 41, 9, 56, 21, 63, 71, 15, 100, 54, 81, 81]
[155, 125, 115, 125, 169, 95, 109, 95, 161, 85, 21, 115, 45, 129, 145, 33, 203, 111, 165, 165]
Standard Deviation 47.387651067384745
```

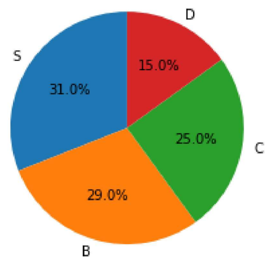
(4) For a given data of heights of a class, the heights of 15 students are recorded as 167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 167, 169, and 172. Develop an application that computes; explore if there are any packages supported in your platform that depicts these measures / their calculations of central tendency in a visual form for ease of understanding. a. Mean height of the student b. Median and Mode of the sample space c. Standard deviation d. Measure of skewness. [(Mean-Mode)/standard deviation]

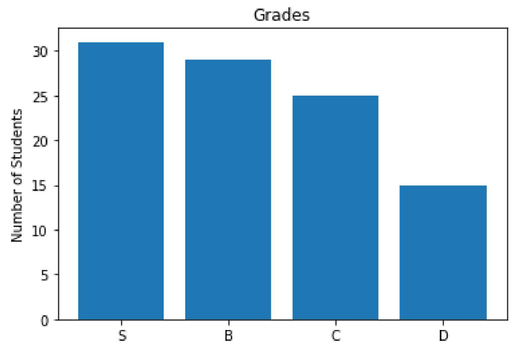
```
import statistics
import matplotlib.pyplot as plt

sample = [167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 167, 169, 172]

mean = statistics.mean(sample)
median = statistics.median(sample)
mode = statistics.mode(sample)
stddev = statistics.stdev(sample)
skewness = (mean - mode)/stddev

print(sample)
print("mean = ",mean,"\nmedian = ",median,"\nmode = ",mode,"\nstddev = ",stddev,"\nskewness = ",skewness)

sample.sort()
plt.bar(x=range(15),height=sample)
plt.show()
```

```
[167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 167, 169, 172]
mean =  168.76333333333332
median =  167.65
mode =  167
stddev =  2.606617037647145
skewness =  0.6764834679838465
```

(5) In Analytics and Systems of Bigdata course, for a class of 100 students, around 31 students secured 'S' grade, 29 secured 'B' grade, 25 'C' grades, and rest of them secured 'D' grades. If the range of each grade is 15 marks. (S for 85 to 100 marks, A for 70 to 85 ...). Develop an application that represents the above data: using Pie and Bar graphs.

```
import numpy as np
import pandas as pd #to work with dataframes
import matplotlib.pyplot as plt

data = {
    'Grades': ['S', 'B', 'C', 'D'],
    'Count' : [31,29,25,15]
    }
df = pd.DataFrame(data, columns = ['Grades', 'Count'])


# Create a pie chart
plt.pie(df['Count'],labels=df['Grades'], autopct='%1.1f%%', startangle=90)    # autopct enables you to display the percent value
plt.show()

# Create a bar graph
x_pos = np.arange(len(df['Grades']))    # arange() function is used to get evenly spaced values within a given interval.
plt.bar(x_pos, df['Count'], align='center')
plt.xticks(x_pos, df['Grades'])
plt.ylabel('Number of Students')
plt.title('Grades')
plt.show()
```

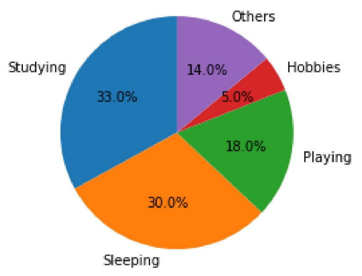




(6) On a given day (average basis), a student is observed to spend 33% of time in studying, 30% in sleeping, 18% in playing, 5% for hobby activities, and rest for spending with friends and family. Plot a pie chart showing his daily activities.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

data = {
    'Activity'   : ['Studying', 'Sleeping', 'Playing', 'Hobbies','Others'],
    'Time Spent' : [33,30,18,5,14]
    }
```

```
df = pd.DataFrame(data, columns = ['Activity', 'Time Spent'])


plt.pie(df['Time Spent'],labels=df['Activity'], autopct='%1.1f%%', startangle=90)
plt.show()
```



(7) Develop an application (absolute grader) that accepts marks scored by 20 students in ASBD course (as a split up of three: Mid Sem (30), End Sem (50) and Assignments(20). Compute the total and use it to grade the students following absolute grading: >=90 – S ; >=80 – A and so on till D. Compute the Class average for total marks in the course and 50% of class average would be fixed as the cut off for E. Generate a frequency table for the grades as well (Table displaying the grades and counts of them).

```
import statistics

MidSem = [10,8,5,21,28,11,14,29,27,29,18,25,26,17,19,23,24,23,21,15]
EndSem = [24,44,12,34,39,37,43,20,25,28,48,45,28,30,46,33,31,29,47,49]
Assignment = [5,7,2,18,12,8,19,14,10,10,16,15,13,17,17,19,13,14,13,11]
Total = []

for i in range(20):
    Total.append(MidSem[i]+EndSem[i]+Assignment[i])

avg = statistics.mean(Total)
Grade = []
print("Class average is:", avg, end="\n\n")
print("Student \tTotal Marks \tGrades")

# deciding grade
for i in range(20):
    if Total[i] >= 90:
        Grade.append('S')
    elif Total[i] >= 80:
        Grade.append('A')
    elif Total[i] >= 70:
        Grade.append('B')
    elif Total[i] >= 60:
        Grade.append('C')
    elif Total[i] >= 50:
        Grade.append('D')
    elif Total[i] >= 0.5*avg:
        Grade.append('E')
    else:
        Grade.append('U')
    print(i+1,'\t\t',Total[i],'\t\t',Grade[i])


# counting frequency
freq = {'S':0,'A':0,'B':0,'C':0,'D':0,'E':0,'U':0}
for elem in Grade:
    freq[elem] += 1
print("\nGrade \tFrequency")
for i in freq:
    print(i, "\t", freq[i])
```

```
Class average is: 66.9

Student         Total Marks     Grades
1                39              E
2                59              D
3                19              U
4                73              B
5                79              B
```

```
6               56              D
7               76              B
8               63              C
9               62              C
10              67              C
11              82              A
12              85              A
13              67              C
14              64              C
15              82              A
16              75              B
17              68              C
18              66              C
19              81              A
20              75              B

Grade   Frequency
S        0
A        4
B        5
C        7
D        2
E        1
U        1
```

(8) Extend the application developed in (7) to support relative grading which uses the class average (mean) and standard deviation to compute the cutoffs for various grades as opposed to fixing them statically; you can refer the sample grader (excel sheet) attached to understand the formulas for fixing the cutoffs; the grader would involve, mean, standard deviation, max mark, passed students data mean, etc. Understand the excel grader thoroughly before you try mimicking such an application in your development platform.

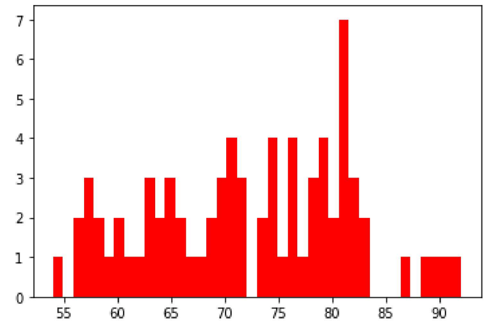
Formulas Required for Relative Grading: • Passing Minimum: 50% of class average. (Minimum marks for passing) • X= Passing Students' Mean-Passing Minimum. • S_cutoff = Max_Mark – 0.1 *(Max_Mark-Passing Students Mean) • Y= S_cutoff – Passing Students Mean • A_cutoff = Passing Students Mean + Y * (5/8) • B_cutoff = Passing Students Mean + Y * (2/8) • C_cutoff = Passing Students Mean - X * (2/8) • D_cutoff = Passing Students Mean - X * (5/8) • E_cutoff = Passing Minimum For the output data generated from Q7 and Q8, apply conditional formatting grade wise and highlight those who failed in red. (use minimum three-color codes)

```python
import statistics

mid_sem = [10,8,5,21,28,11,14,29,27,29,18,25,26,17,19,23,24,23,21,15]
end_sem = [24,44,12,34,39,37,43,20,25,28,48,45,28,30,46,33,31,29,47,49]
assignment = [5,7,2,18,12,8,19,14,10,10,16,15,13,17,17,19,13,14,13,11]
total = []

for i in range(20):
    total.append(mid_sem[i]+end_sem[i]+assignment[i])

avg = statistics.mean(total)
passing_min = avg*0.5

passing_students_total_marks = 0
passing_student_number = 0
for i in range(20):
    if total[i] >= passing_min:
        passing_students_total_marks += total[i]
        passing_student_number += 1
passing_student_mean = passing_students_total_marks/passing_student_number

X = passing_student_mean - passing_min
max_mark = max(total)
S_cutoff = max_mark -0.1*(max_mark - passing_student_mean)
Y = S_cutoff - passing_student_mean
A_cutoff = passing_student_mean + Y*(5/8)
B_cutoff = passing_student_mean + Y*(2/8)
C_cutoff = passing_student_mean - X*(2/8)
D_cutoff = passing_student_mean - X*(5/8)
E_cutoff = passing_min


Grade = []
print("Class average is:", avg, end="\n\n")
print("Student \tTotal Marks \tGrades")

# deciding grade
for i in range(20):
    if total[i] >= S_cutoff:
        Grade.append('S')
    elif total[i] >= A_cutoff:
        Grade.append('A')
    elif total[i] >= B_cutoff:
        Grade.append('B')
    elif total[i] >= C_cutoff:
        Grade.append('C')
    elif total[i] >= D_cutoff:
        Grade.append('D')
    elif total[i] >= E_cutoff:
        Grade.append('E')
    else:
        Grade.append('U')
    print(i+1,'\t\t',total[i],'\t\t',Grade[i])


# counting frequency
freq = {'S':0,'A':0,'B':0,'C':0,'D':0,'E':0,'U':0}
for elem in Grade:
    freq[elem] += 1
print("\nGrade \tFrequency")
for i in freq:
    print(i, "\t", freq[i])
```

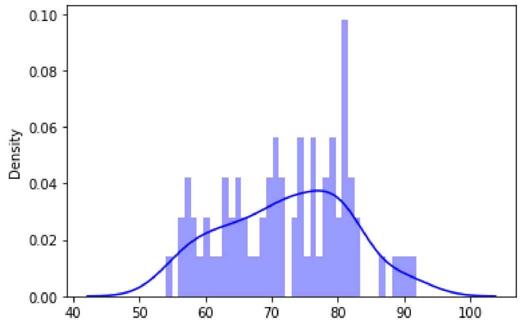
```
Class average is: 66.9

Student         Total Marks     Grades
1                39              E
2                59              D
3                19              U
4                73              B
5                79              A
6                56              D
7                76              B
8                63              C
9                62              C
10               67              C
11               82              A
12               85              S
13               67              C
14               64              C
15               82              A
16               75              B
17               68              C
18               66              C
19               81              A
20               75              B

Grade   Frequency
S        1
A        4
B        4
C        7
D        2
E        1
U        1
```

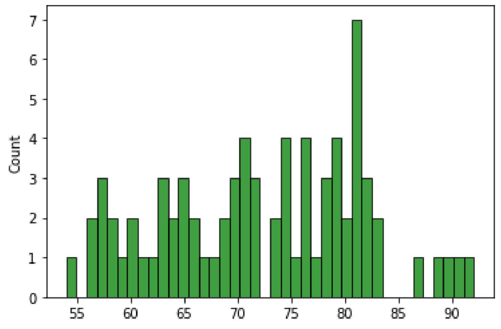
(9) Consider the following sample of weights for 45 individuals: 79 71 89 57 76 64 82 82 67 80 81 65 73 79 79 60 58 83 74 68 78 80 78 81 76 65 70 76 58 82 59 73 72 79 87 63 74 90 69 70 83 76 61 66 71 60 57 81 57 65 81 78 77 81 81 63 71 66 56 62 75 64 74 74 70 71 56 69 63 72 81 54 72 91 92. For the above data generates histograms and depict them using packages in your platform. Explore the different types of histograms available and test drive the types supported in your platform

```
import matplotlib.pyplot as plt
import seaborn as sns


weights = [79,71,89,57,76,64,82,82,67,80,81,65,73,79,79,60,58,83,74,
           68,78,80,78,81,76,65,70,76,58,82,59,73,72,79,87,63,74,90,
           69,70,83,76,61,66,71,60,57,81,57,65,81,78,77,81,81,63,71,
           66,56,62,75,64,74,74,70,71,56,69,63,72,81,54,72,91,92]

plt.hist(x=weights,bins= 40,color='red')
plt.show()
sns.distplot(a=weights,bins=40,color="Blue")
plt.show()
sns.histplot(data=weights, bins=40, color="Green")
plt.show()
```



```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `d
  warnings.warn(msg, FutureWarning)
```





Colab paid products - Cancel contracts here

✓ 2s completed at 2:47 PM