ASBD Lab - 1

In [64]:

```
import numpy as np
from scipy import stats as st
import random
from tabulate import tabulate
import matplotlib.pyplot as plt
from matplotlib import colors

random.seed(10)
```

Question 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.

In [53]:

```python

tallyDict = {
    1: "|",
    2: "||",
    3: "|||",
    4: "||||",
    5: "||||/",
}


result = []
for i in range(10):
    finalString = ""
    number = random.randint(0,50)
    countFive = number//5
    countRemaining = number%5

    if(countFive):
        finalString = tallyDict[5]*countFive
    if(countRemaining):
        finalString = finalString + tallyDict[countRemaining]

    result.append([finalString, number])


head = ["Tally", "Frequency"]
print(tabulate(result, headers=head, tablefmt='fancy_grid', showindex=T

```

```
╒════╤══════════════════════════════════════╤═════════════╕
│    │ Tally                                │   Frequency │
╞════╪══════════════════════════════════════╪═════════════╡
│  0 │ ||||/||||/||||/||||/||||/||||/||||/| │          36 │
├────┼──────────────────────────────────────┼─────────────┤
│  1 │ ||                                   │           2 │
├────┼──────────────────────────────────────┼─────────────┤
│  2 │ ||||/||||/||||/||||/||||/||          │          27 │
├────┼──────────────────────────────────────┼─────────────┤
│  3 │ ||||/||||/||||/||||/||||/||||/       │          30 │
├────┼──────────────────────────────────────┼─────────────┤
│  4 │ ||||/||||/||||/||||/||||/||||/||||/| │          36 │
├────┼──────────────────────────────────────┼─────────────┤
│  5 │                                      │           0 │
├────┼──────────────────────────────────────┼─────────────┤
│  6 │ ||||/||||/|||                        │          13 │
├────┼──────────────────────────────────────┼─────────────┤
│  7 │ ||||/||||/||||/||||/||||/||||        │          29 │
├────┼──────────────────────────────────────┼─────────────┤
│  8 │ ||||/||||/||||/||||/||||/||||/|      │          31 │
├────┼──────────────────────────────────────┼─────────────┤
│  9 │ ||||/||||/||||/||                    │          17 │
╘════╧══════════════════════════════════════╧═════════════╛
```

Question 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.

In [54]:

```python


rollnumString = "CSE20D0"
resultTable = []
marksList = []
# generate 18 roll numbers and their marks
for i in range(1,19):
    newStudent = rollnumString + str(i)
#     odd
    if(i&1):
        marks = 25+((i+7)%10)
    else:  #even
        marks = 25+((i+8)%10)

    resultTable.append([newStudent, marks])
    marksList.append(marks)



resultTable.append(["Mean", np.mean(marksList)])
resultTable.append(["Median", np.median(marksList)])

head = ["Student", "Marks"]
print(tabulate(resultTable, headers=head, tablefmt='fancy_grid', showin

```

```
╒════╤═══════════╤═════════╕
│    │ Student   │   Marks │
╞════╪═══════════╪═════════╡
│  0 │ CSE20D01  │ 33      │
├────┼───────────┼─────────┤
│  1 │ CSE20D02  │ 25      │
├────┼───────────┼─────────┤
│  2 │ CSE20D03  │ 25      │
├────┼───────────┼─────────┤
│  3 │ CSE20D04  │ 27      │
├────┼───────────┼─────────┤
│  4 │ CSE20D05  │ 27      │
├────┼───────────┼─────────┤
│  5 │ CSE20D06  │ 29      │
├────┼───────────┼─────────┤
│  6 │ CSE20D07  │ 29      │
├────┼───────────┼─────────┤
│  7 │ CSE20D08  │ 31      │
├────┼───────────┼─────────┤
│  8 │ CSE20D09  │ 31      │
├────┼───────────┼─────────┤
│  9 │ CSE20D010 │ 33      │
├────┼───────────┼─────────┤
│ 10 │ CSE20D011 │ 33      │
├────┼───────────┼─────────┤
│ 11 │ CSE20D012 │ 25      │
├────┼───────────┼─────────┤
│ 12 │ CSE20D013 │ 25      │
├────┼───────────┼─────────┤
│ 13 │ CSE20D014 │ 27      │
├────┼───────────┼─────────┤
│ 14 │ CSE20D015 │ 27      │
├────┼───────────┼─────────┤
│    │           │         │
│ 15 │ CSE20D016 │ 29      │
├────┼───────────┼─────────┤
│ 16 │ CSE20D017 │ 29      │
├────┼───────────┼─────────┤
│ 17 │ CSE20D018 │ 31      │
├────┼───────────┼─────────┤
│ 18 │ Mean      │ 28.6667 │
├────┼───────────┼─────────┤
│ 19 │ Median    │ 29      │
└────┴───────────┴─────────┘
```

Question 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).

In [55]:

```python
resultTable = []
for i in range(20):
    X = random.randint(6,200)
    Y = 2*X+3
    resultTable.append([X,Y])



resultTable.append(["Standard Deviation",  np.std(resultTable, ddof=1)]

head = ["X", "Y"]
print(tabulate(resultTable, headers=head, tablefmt='fancy_grid', showin
```

```
╒════╤════════════════════╤══════════╕
│    │ X                  │        Y │
╞════╪════════════════════╪══════════╡
│  0 │ 173                │ 349      │
├────┼────────────────────┼──────────┤
│  1 │ 47                 │  97      │
├────┼────────────────────┼──────────┤
│  2 │ 14                 │  31      │
├────┼────────────────────┼──────────┤
│  3 │ 139                │ 281      │
├────┼────────────────────┼──────────┤
│  4 │ 131                │ 265      │
├────┼────────────────────┼──────────┤
│  5 │ 89                 │ 181      │
├────┼────────────────────┼──────────┤
│  6 │ 25                 │  53      │
├────┼────────────────────┼──────────┤
│  7 │ 69                 │ 141      │
├────┼────────────────────┼──────────┤
│  8 │ 196                │ 395      │
├────┼────────────────────┼──────────┤
│  9 │ 98                 │ 199      │
├────┼────────────────────┼──────────┤
│ 10 │ 17                 │  37      │
├────┼────────────────────┼──────────┤
│ 11 │ 113                │ 229      │
├────┼────────────────────┼──────────┤
│ 12 │ 41                 │  85      │
├────┼────────────────────┼──────────┤
│ 13 │ 160                │ 323      │
├────┼────────────────────┼──────────┤
│ 14 │ 96                 │ 195      │
├────┼────────────────────┼──────────┤
│ 15 │ 103                │ 209      │
├────┼────────────────────┼──────────┤
│ 16 │ 113                │ 229      │
├────┼────────────────────┼──────────┤
│ 17 │ 78                 │ 159      │
├────┼────────────────────┼──────────┤
│ 18 │ 178                │ 359      │
├────┼────────────────────┼──────────┤
│ 19 │ 73                 │ 149      │
├────┼────────────────────┼──────────┤
│ 20 │ Standard Deviation │  98.5298 │
╘════╧════════════════════╧══════════╛
```

Question 4

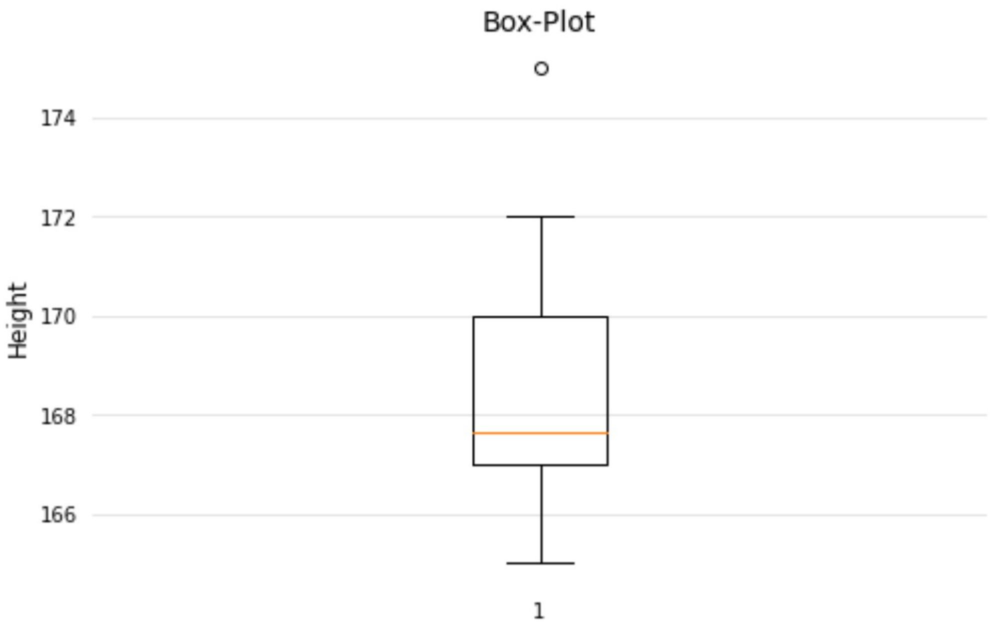
For a given data of heights of a class, the heights of 15 students are recorded as 167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5, 170, 167, 169, and 172. Develop an application that computes; explore if there are any packages supported in your platform that depicts these measures / their 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]

In [56]:

```
heights = [167.65, 167, 172, 175, 165, 167, 168, 167, 167.3, 170, 167.5
resultTable = []
resultTable.append(["Mean", np.mean(heights)])
resultTable.append(["Median", np.median(heights)])
resultTable.append(["Mode", st.mode(heights)[0][0]])
resultTable.append(["Standard Deviation",  np.std(heights, ddof=1)])
resultTable.append(["Measure of Skewness",  ((np.mean(heights)-st.mode(

head = ["Measure", "Value"]
print(tabulate(resultTable, headers=head, tablefmt='fancy_grid', showin


plt.figure(figsize=(8,5))
plt.boxplot(heights)
plt.grid(axis ='y',alpha=0.4)
plt.tick_params(left = False, bottom=False)
plt.ylabel("Height", fontsize=12)
plt.title("Box-Plot", fontsize=14)

for spine in plt.gca().spines.values():
    spine.set_visible(False)
plt.show()
```

```
╒════╤═════════════════════╤════════════╕
│    │ Measure             │      Value │
╞════╪═════════════════════╪════════════╡
│  0 │ Mean                │ 168.763    │
├────┼─────────────────────┼────────────┤
│  1 │ Median              │ 167.65     │
├────┼─────────────────────┼────────────┤
│  2 │ Mode                │ 167        │
├────┼─────────────────────┼────────────┤
│  3 │ Standard Deviation  │   2.60662  │
├────┼─────────────────────┼────────────┤
│  4 │ Measure of Skewness │   0.676483 │
╘════╧═════════════════════╧════════════╛
```

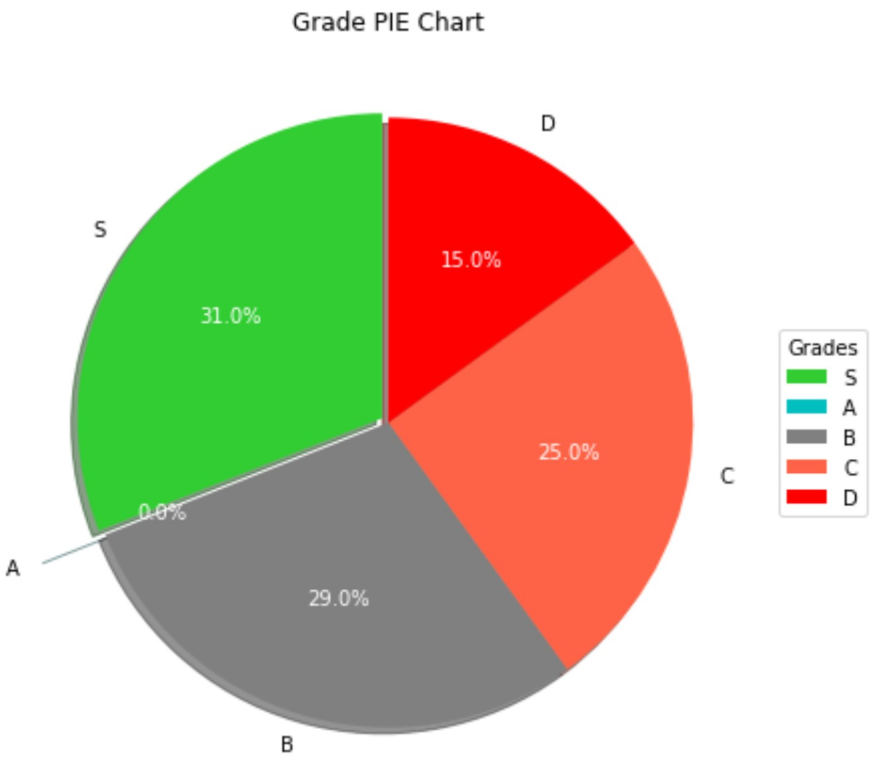


Question 5

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

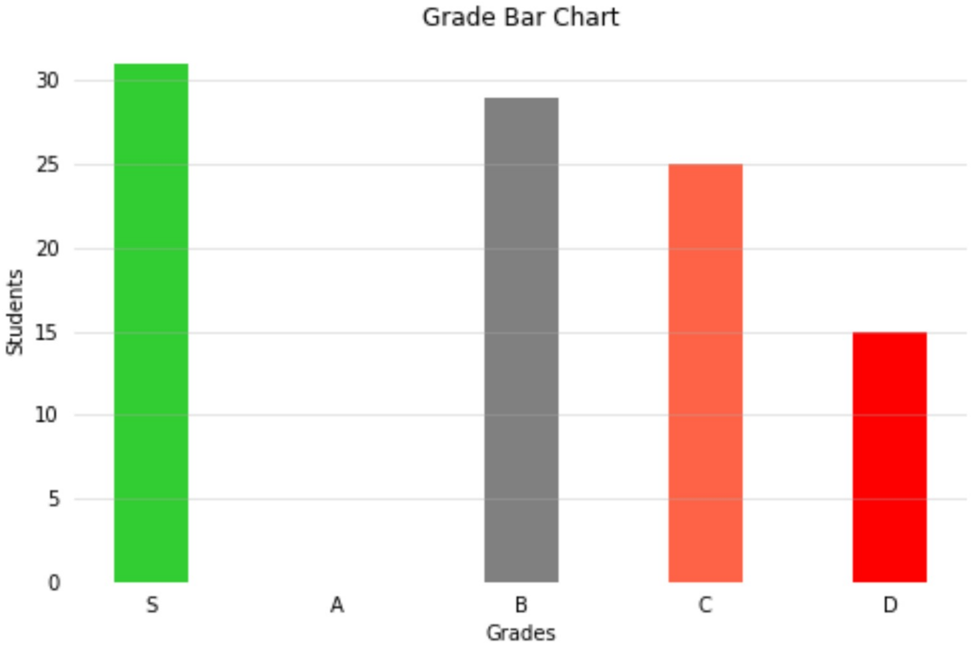
In [58]:

```python
grades = ('S', 'A', 'B', 'C', 'D')
student_count = (31, 0, 29, 25, 15)
explodes = (0.025, 0.2, 0, 0, 0)
colors = ("limegreen", "c", "grey", "tomato", "red")

fig, ax = plt.subplots(figsize=(10,7))
_, _, autotexts = ax.pie(student_count,
     labels = grades,
     explode = explodes,
      shadow = True,
      colors = colors,
      startangle = 90,
      autopct='%1.1f%%'
     )

for autotext in autotexts:
    autotext.set_color('white')

ax.legend(grades,
         title="Grades",
         loc="center left",
         bbox_to_anchor=(1,0,0.5,1))

ax.set_title("Grade PIE Chart")
plt.show()
```



In [59]:

```
grades = ('S', 'A', 'B', 'C', 'D')
student_count = (31, 0, 29, 25, 15)
colors = ("limegreen", "c", "grey", "tomato", "red")

fig, ax = plt.subplots(figsize=(8,5))

ax.bar(grades,
       student_count,
       color=colors,
       width=0.4
      )

ax.set_xlabel("Grades")
ax.set_ylabel("Students")
ax.set_title("Grade Bar Chart")

plt.grid(axis ='y',alpha=0.4)
plt.tick_params(left = False, bottom=False)

for spine in plt.gca().spines.values():
    spine.set_visible(False)

plt.show()
```

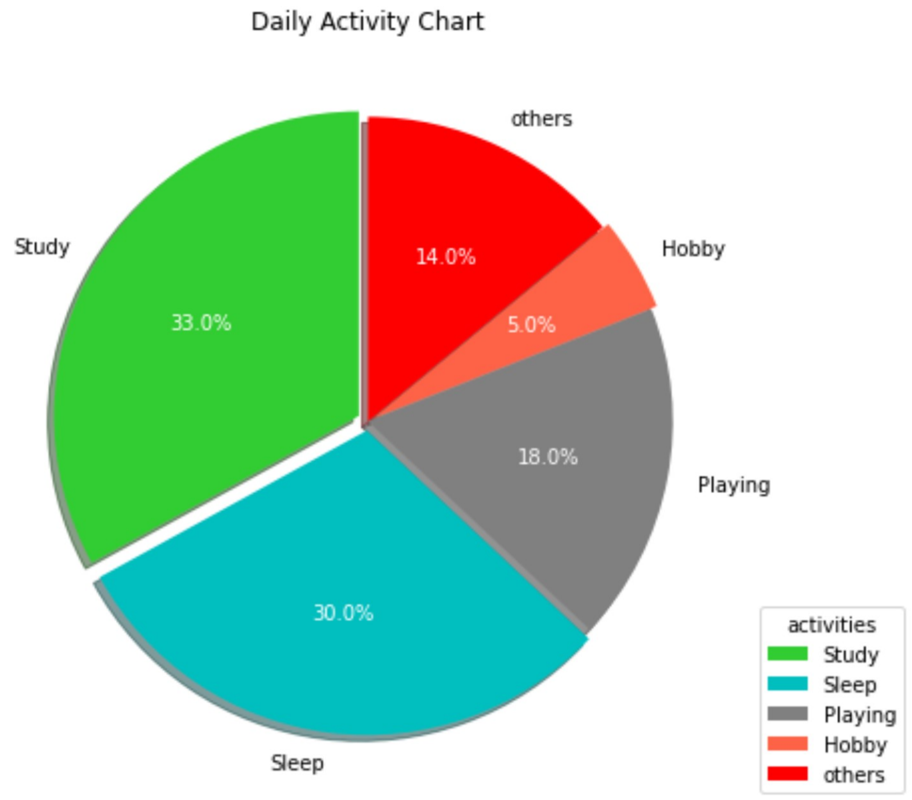


Question 6

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.

In [60]:

```python
activities = ('Study', 'Sleep', 'Playing', 'Hobby', 'others')
time = (33, 30, 18, 5, 14)
explodes = (0.035, 0.03, 0, 0.02, 0)
colors = ("limegreen", "c", "grey", "tomato", "red")

fig, ax = plt.subplots(figsize=(10,7))
_, _, autotexts = ax.pie(time,
     labels = activities,
     explode = explodes,
      shadow = True,
      colors = colors,
      startangle = 90,
      autopct='%1.1f%%'
     )

for autotext in autotexts:
    autotext.set_color('white')

ax.legend(activities,
         title="activities",
         loc="lower left",
         bbox_to_anchor=(1,0,0.5,1))

ax.set_title("Daily Activity Chart")
plt.show()
```



Question 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).

In [61]:

```
student_list = [
    ["CSE20B001",20],
    ["CSE20B002",30],
    ["CSE20B003",40],
    ["CSE20B004",78],
    ["CSE20B005",70],
    ["CSE20B006",65],
    ["CSE20B007",65],
    ["CSE20B008",88],
    ["CSE20B009",87],
    ["CSE20B010",78],
    ["CSE20B011",76],
    ["CSE20B012",68],
    ["CSE20B013",45],
    ["CSE20B014",69],
    ["CSE20B015",93],
    ["CSE20B016",23],
    ["CSE20B017",45],
    ["CSE20B018",78],
    ["CSE20B019",97],
    ["CSE20B020",78]]


class_avg = 0;
for each in student_list:
    class_avg += each[1]

class_avg = class_avg/20
E_limit = class_avg*0.5


grade_dict = {
    'S': 0,
    'A': 0,
    'B': 0,
    'C': 0,
    'D': 0,
    'E': 0,
    'F': 0
}

for each in student_list:
    if(each[1] >= 90):
        grade_dict['S'] += 1
    elif(each[1] >= 80):
         grade_dict['A'] += 1
    elif(each[1] >= 70):
         grade_dict['B'] += 1
    elif(each[1] >= 60):
         grade_dict['C'] += 1
    elif(each[1] >= 50):
         grade_dict['D'] += 1
    elif(each[1] >= E_limit):
         grade_dict['E'] += 1
    else:
        grade_dict['F'] += 1

result_table = list(map(list, grade_dict.items()))
result_table.append(['Class Avg', class_avg])
```

```

head = ["Grade", "Student Count"]
print(tabulate(result_table, headers=head, tablefmt='fancy_grid', showi
```

```
╒════╤═══════════╤═══════════════╕
│    │ Grade     │ Student Count │
╞════╪═══════════╪═══════════════╡
│  0 │ S         │          2    │
├────┼───────────┼───────────────┤
│  1 │ A         │          2    │
├────┼───────────┼───────────────┤
│  2 │ B         │          6    │
├────┼───────────┼───────────────┤
│  3 │ C         │          4    │
├────┼───────────┼───────────────┤
│  4 │ D         │          0    │
├────┼───────────┼───────────────┤
│  5 │ E         │          3    │
├────┼───────────┼───────────────┤
│  6 │ F         │          3    │
├────┼───────────┼───────────────┤
│  7 │ Class Avg │         64.65 │
╘════╧═══════════╧═══════════════╛
```

Question 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.

In [62]:

```
student_list = [
    ["CSE20B001",20],
    ["CSE20B002",30],
    ["CSE20B003",40],
    ["CSE20B004",78],
    ["CSE20B005",70],
    ["CSE20B006",65],
    ["CSE20B007",65],
    ["CSE20B008",88],
    ["CSE20B009",87],
    ["CSE20B010",78],
    ["CSE20B011",76],
    ["CSE20B012",68],
    ["CSE20B013",45],
    ["CSE20B014",69],
    ["CSE20B015",93],
    ["CSE20B016",23],
    ["CSE20B017",45],
    ["CSE20B018",78],
    ["CSE20B019",97],
    ["CSE20B020",78]]


class_avg = 0
max_mark = 0
for each in student_list:
    if(each[1] > max_mark):
        max_mark = each[1]

    class_avg += each[1]

class_avg = class_avg/20
passing_minimum = class_avg*0.5

passing_student_mean = 0
passing_student_count = 0
for each in student_list:
    if(each[1] > passing_minimum):
        passing_student_mean += each[1]
        passing_student_count += 1

passing_student_mean = passing_student_mean / passing_student_count
X = passing_student_mean - passing_minimum
S_cutoff = max_mark - 0.1*(max_mark-passing_student_mean)

Y = S_cutoff - passing_student_mean
A_cutoff = passing_student_mean + Y*0.625
B_cutoff = passing_student_mean + Y*0.25
C_cutoff = passing_student_mean - X*0.25
D_cutoff = passing_student_mean - X*0.625
E_cutoff = passing_minimum


grade_dict = {
    'S': 0,
    'A': 0,
    'B': 0,
    'C': 0,
    'D': 0,
    'E': 0,
    'F': 0
}

for each in student_list:
    if(each[1] >= S_cutoff):
        grade_dict['S'] += 1
    elif(each[1] >= A_cutoff):
         grade_dict['A'] += 1
    elif(each[1] >= B_cutoff):
         grade_dict['B'] += 1
    elif(each[1] >= C_cutoff):
         grade_dict['C'] += 1
    elif(each[1] >= D_cutoff):
         grade_dict['D'] += 1
    elif(each[1] >= E_cutoff):
         grade_dict['E'] += 1
    else:
        grade_dict['F'] += 1


result_table = list(map(list, grade_dict.items()))
result_table.append(['Passing Minimum', passing_minimum])

head = ["Grade", "Student Count"]
print(tabulate(result_table, headers=head, tablefmt='fancy_grid', showi
```

```
╒════╤═════════════════╤═════════════════╕
│    │ Grade           │   Student Count │
╞════╪═════════════════╪═════════════════╡
│  0 │ S               │           1     │
├────┼─────────────────┼─────────────────┤
│  1 │ A               │           3     │
├────┼─────────────────┼─────────────────┤
│  2 │ B               │           4     │
├────┼─────────────────┼─────────────────┤
│  3 │ C               │           6     │
├────┼─────────────────┼─────────────────┤
│  4 │ D               │           0     │
├────┼─────────────────┼─────────────────┤
│  5 │ E               │           3     │
├────┼─────────────────┼─────────────────┤
│  6 │ F               │           3     │
├────┼─────────────────┼─────────────────┤
│  7 │ Passing Minimum │          32.325 │
╘════╧═════════════════╧═════════════════╛
```

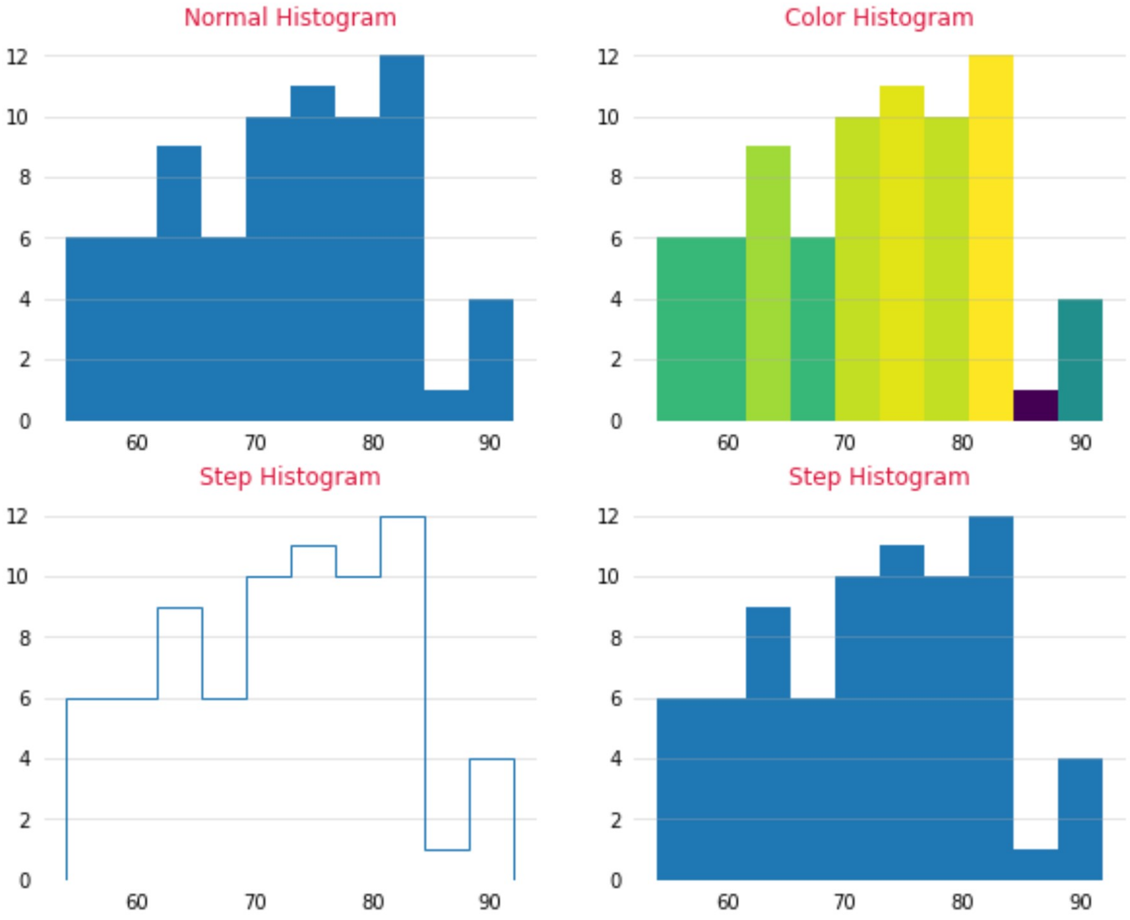
Question 9

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

In [65]:

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

def plotGraph():
    plt.hist(data)
    plt.title("Normal Histogram" , color='crimson', fontsize=12)
    plt.grid(axis ='y',alpha=0.4)

    plt.tick_params(left = False, bottom=False)
    for spine in plt.gca().spines.values():
        spine.set_visible(False)


def plotStepGraph():
    plt.hist(data, histtype='step')
    plt.title("Step Histogram" , color='crimson', fontsize=12)
    plt.grid(axis ='y',alpha=0.4)

    plt.tick_params(left = False, bottom=False)
    for spine in plt.gca().spines.values():
        spine.set_visible(False)

def plotBarHist():
    plt.hist(data, histtype='bar')
    plt.title("Step Histogram" , color='crimson', fontsize=12)
    plt.grid(axis ='y',alpha=0.4)

    plt.tick_params(left = False, bottom=False)
    for spine in plt.gca().spines.values():
        spine.set_visible(False)


def plotColotGraph():
    N, bins, patches = plt.hist(data)
    plt.title("Color Histogram" , color='crimson', fontsize=12)
    plt.grid(axis ='y',alpha=0.4)

    plt.tick_params(left = False, bottom=False)
    for spine in plt.gca().spines.values():
        spine.set_visible(False)

    fracs = ((N**(1 / 5)) / N.max())
    norm = colors.Normalize(fracs.min(), fracs.max())

    for thisfrac, thispatch in zip(fracs, patches):
        color = plt.cm.viridis(norm(thisfrac))
        thispatch.set_facecolor(color)


plt.figure(figsize=(10,8))
plt.subplot(2, 2, 1)
plotGraph()
plt.subplot(2, 2, 2)
plotColotGraph()
plt.subplot(2, 2, 3)
plotStepGraph()
```

```python
plt.subplot(2, 2, 4)
plotBarHist()

# plt.show()
```



In [ ]:

In [ ]: