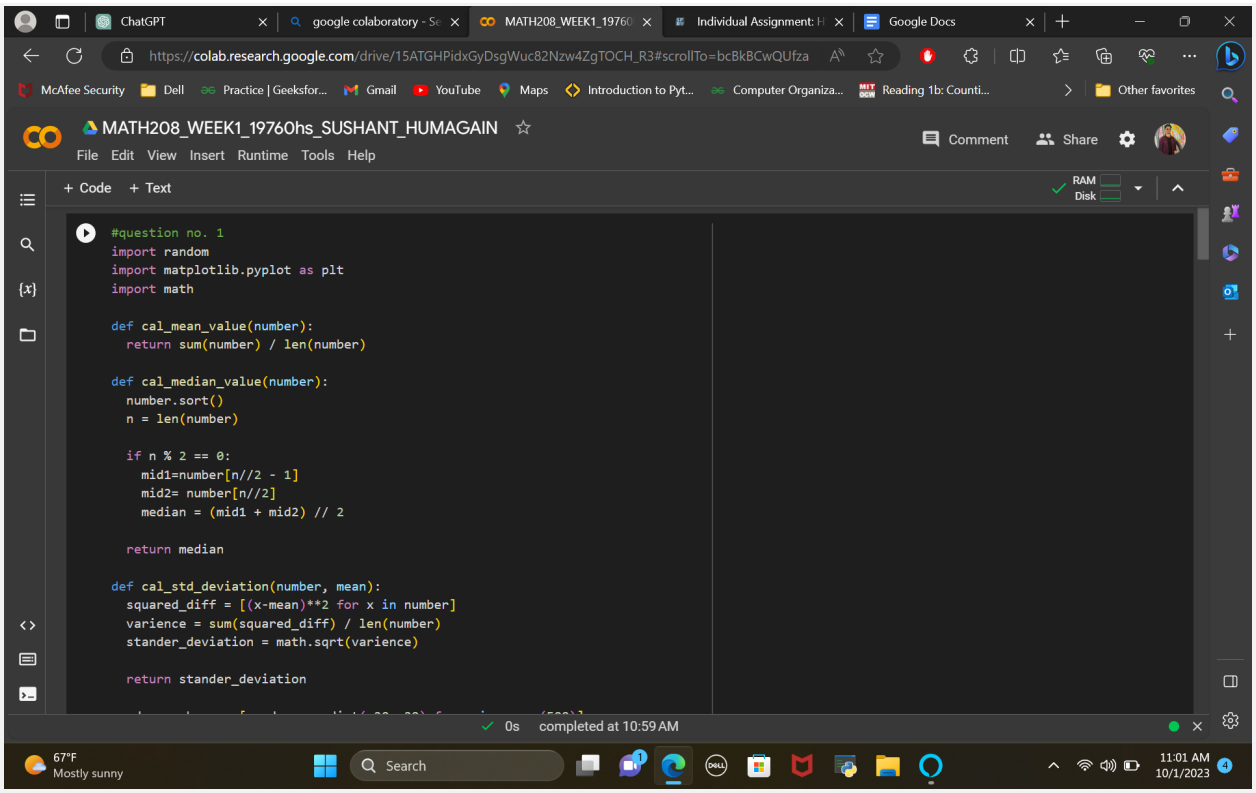
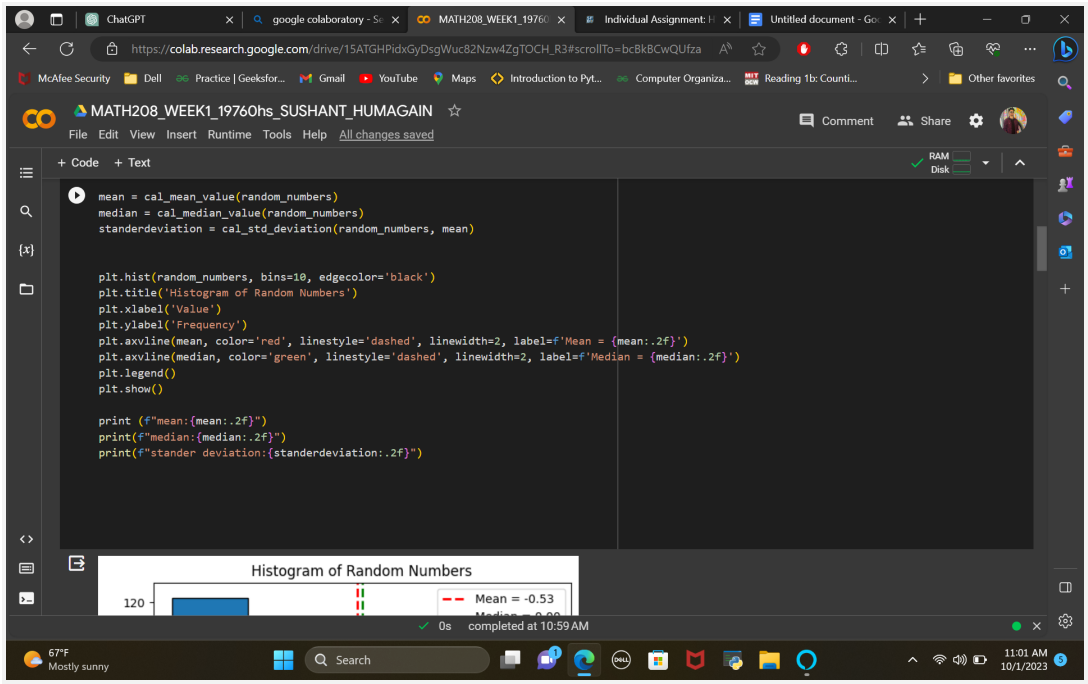
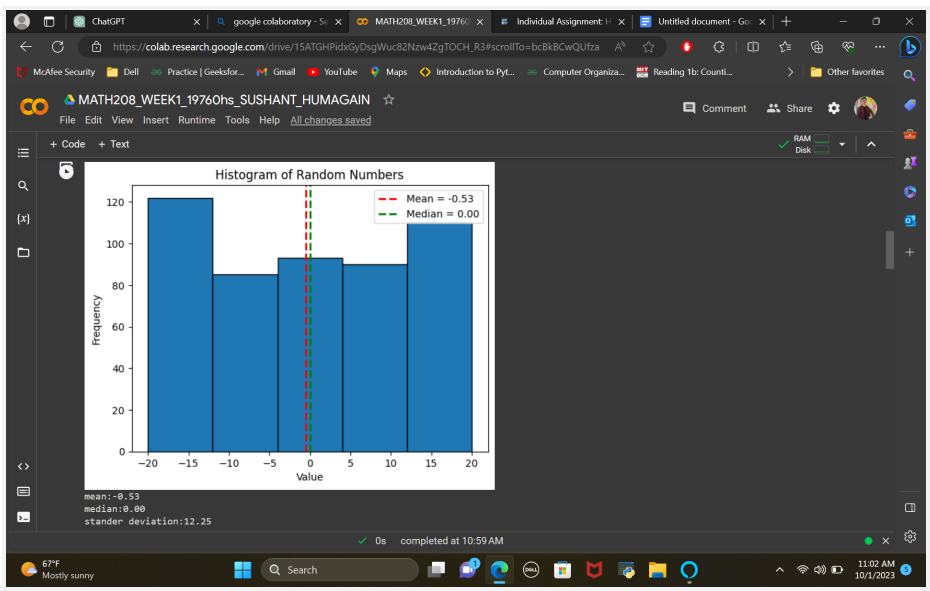
#question no 1.

Write the program in any computer language, Python preferred to create 500 random numbers from -20 to +20 in uniform distribution and find the mean, median and standard deviation. After that, plot the histogram with 10 bins. Notice that the only user defined function can be used to calculate the mean, median and standard deviation, don’t directly call existing function from PytWrite the program in any computer lar

numbers from -20 to +20 in uniform distribution and find the mean, median and standard deviation. After that, plot the histogram with 10 bins. Notice that the only user defined function can be used to calculate the mean, median and standard deviation, don’t directly call existing function from Python library.hon library.







#question no .2

Similar to the above, write the program to create 500 random numbers with mean = 10 and standard deviation = 0.5 in Gaussian distribution and find the mean, median and standard deviation. After that, plot the histogram with 10 bins. Notice that the only user defined function can be used to calculate the mean, median and standard deviation, don't directly call existing function from Python library.

MATH208_WEEK1_19760hs_SUSHANT_HUMAGAIN

```python
#que no.2
import random
import math
import matplotlib.pyplot as plt

def generate_gaussian_numbers(mean, std_deviation, count):
    random_numbers = [random.gauss(mean, std_deviation) for _ in range(count)]
    return random_numbers


def cal_mean_value(numbers):
    return sum(numbers) / len(numbers)

def cal_median_value(numbers):
    numbers.sort()
    n = len(numbers)
    if n % 2 == 0:
        mid1 = numbers[n // 2 - 1]
        mid2 = numbers[n // 2]
        median = (mid1 + mid2) / 2
    else:
        median = numbers[n // 2]
    return median

def cal_std_deviation(numbers, mean):
    squared_diff = [(x - mean) ** 2 for x in numbers]
    variance = sum(squared_diff) / len(numbers)
```

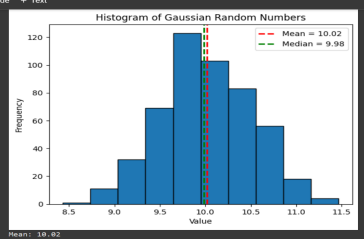
0s completed at 10:59 AM

MATH208_WEEK1_19760hs_SUSHANT_HUMAGAIN

```python
    squared_diff = [(x - mean) ** 2 for x in numbers]
    variance = sum(squared_diff) / len(numbers)
    std_deviation = math.sqrt(variance)
    return std_deviation

mean_value = 10
std_deviation_value = 0.5
random_numbers = generate_gaussian_numbers(mean_value, std_deviation_value, 500)

mean = cal_mean_value(random_numbers)
median = cal_median_value(random_numbers)
std_deviation = cal_std_deviation(random_numbers, mean)

plt.hist(random_numbers, bins=10, edgecolor='black')
plt.title('Histogram of Gaussian Random Numbers')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.axvline(mean, color='red', linestyle='dashed', linewidth=2, label=f'Mean = {mean:.2f}')
plt.axvline(median, color='green', linestyle='dashed', linewidth=2, label=f'Median = {median:.2f}')
plt.legend()
plt.show()

print(f"Mean: {mean:.2f}")
print(f"Median: {median:.2f}")
print(f"Standard Deviation: {std_deviation:.2f}")
```

0s completed at 10:59 AM

MATH208_WEEK1_19760hs_SUSHANT_HUMAGAIN




```
Mean: 10.02
Median: 9.98
Standard Deviation: 0.50
```

0s completed at 10:59 AM

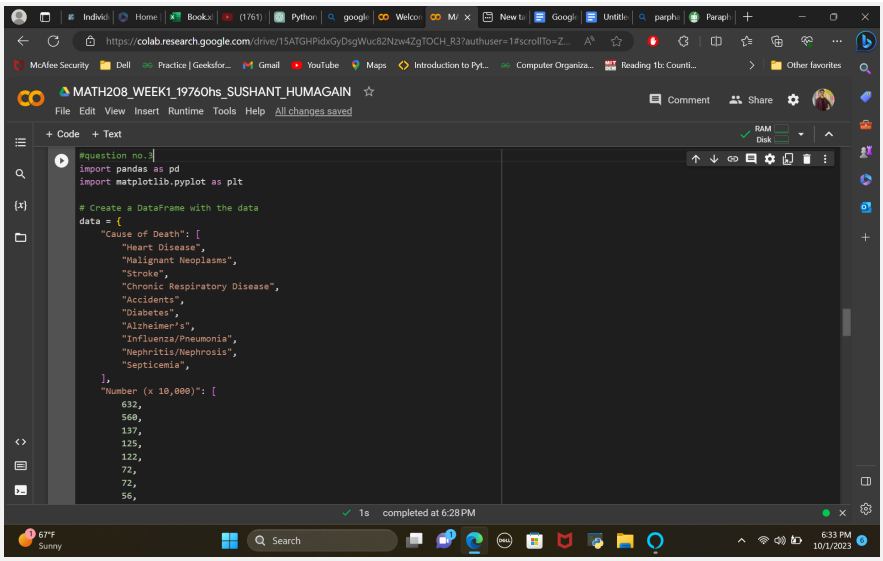
Question no. 3

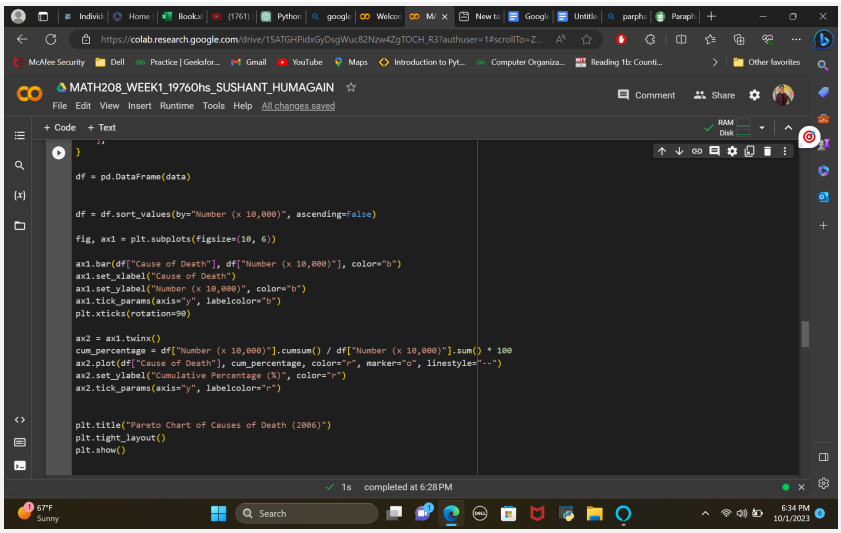
The 10-leading causes of death in the United States during 2006 were listed on the Centers for Disease Control and Prevention website. There are a total of 1,855,610 deaths recorded. Plot the Pareto chart in Python or Excel and explain your results

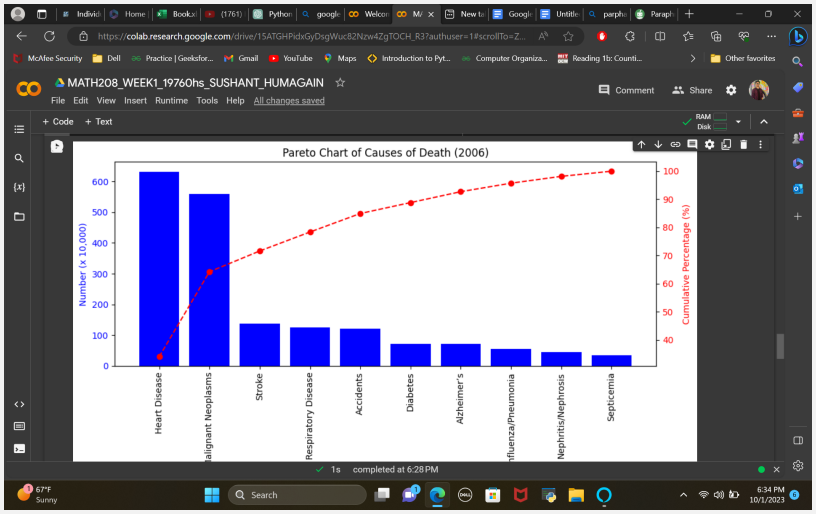
⇒

This code first creates a DataFrame using the given data and then sorts it in descending or line chart for the cumulative percentage on the right y-axis.

The resulting chart will show the causes of death in descending order of frequency, with the deaths. This helps identify the most significant contributors, following the Pareto principle.

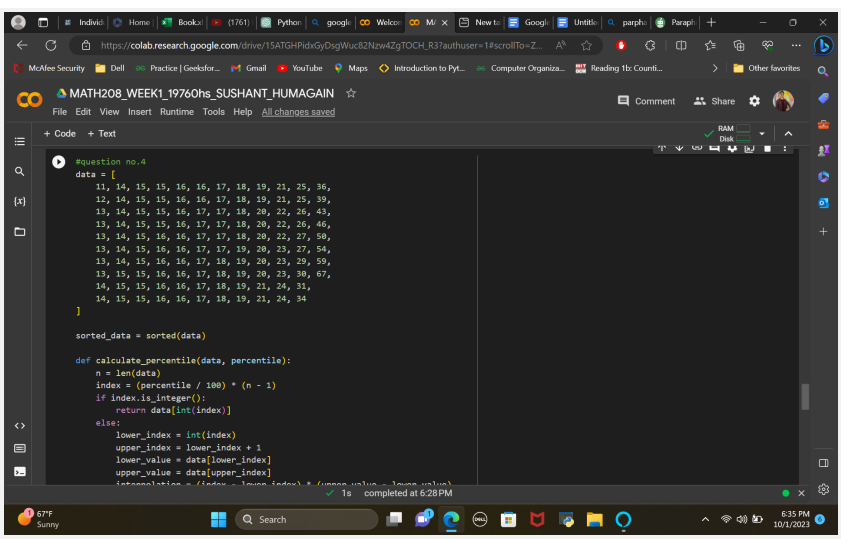






#question no 4

The following data are the ages of 118 known offenders who committed an auto theft last year in Garden City, Michigan. Write the program to find the median, the mode, Q1 and Q3, P10 and P95.



MATH208_WEEK1_19760hs_SUSHANT_HUMAGAIN

```python
    return mode



median = calculate_median(sorted_data)
mode = calculate_mode(data)
q1 = calculate_percentile(sorted_data, 25)
q3 = calculate_percentile(sorted_data, 75)
p10 = calculate_percentile(sorted_data, 10)
p95 = calculate_percentile(sorted_data, 95)

print(f"Median: {median}")
print(f"Mode: {mode}")
print(f"Q1 (1st Quartile): {q1}")
print(f"Q3 (3rd Quartile): {q3}")
print(f"P10 (10th Percentile): {p10}")
print(f"P95 (95th Percentile): {p95}")
```

```
Median: 17.0
Mode: [16]
Q1 (1st Quartile): 15.0
Q3 (3rd Quartile): 20.75
P10 (10th Percentile): 14.0
P95 (95th Percentile): 39.59999999999966
```