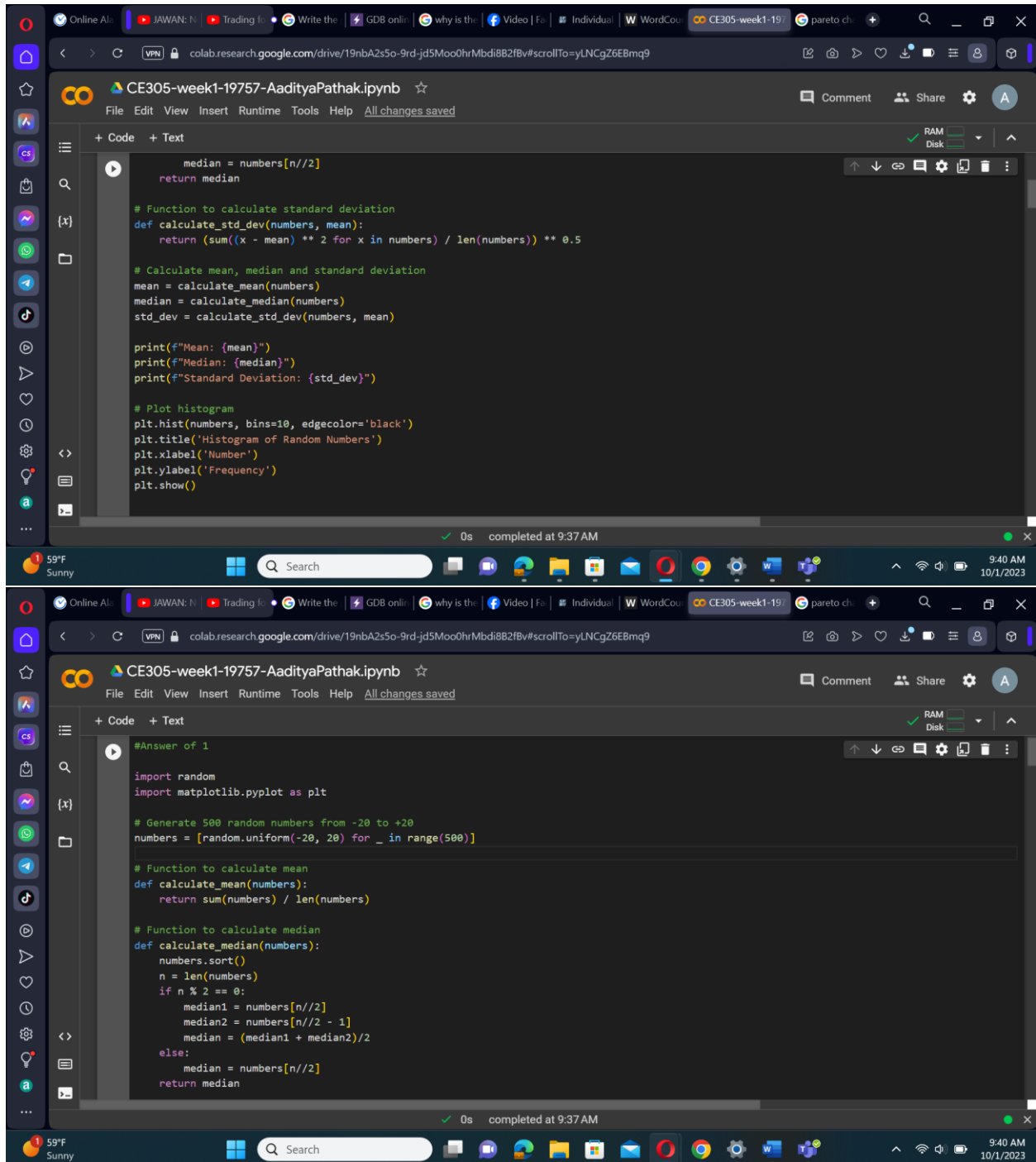


1. Write the program in any computer language, Python preferred to create 500 random numbers from -20 to +20 in uniform distribution and find the mean, median and standard deviation. After that, plot the histogram with 10 bins. Notice that the only user defined function can be used to calculate the mean,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



The image displays two screenshots of a Jupyter Notebook interface, showing the implementation of a program to generate random numbers, calculate statistics, and plot a histogram.

Top Screenshot: The notebook is titled "CE305-week1-19757-AadityaPathak.ipynb". The code defines a function to calculate the median and a function to calculate the standard deviation. It then calculates the mean, median, and standard deviation for a set of 500 random numbers and plots a histogram.

```
median = numbers[n//2]
return median

# Function to calculate standard deviation
def calculate_std_dev(numbers, mean):
    return (sum((x - mean) ** 2 for x in numbers) / len(numbers)) ** 0.5

# Calculate mean, median and standard deviation
mean = calculate_mean(numbers)
median = calculate_median(numbers)
std_dev = calculate_std_dev(numbers, mean)

print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")

# Plot histogram
plt.hist(numbers, bins=10, edgecolor='black')
plt.title('Histogram of Random Numbers')
plt.xlabel('Number')
plt.ylabel('Frequency')
plt.show()
```

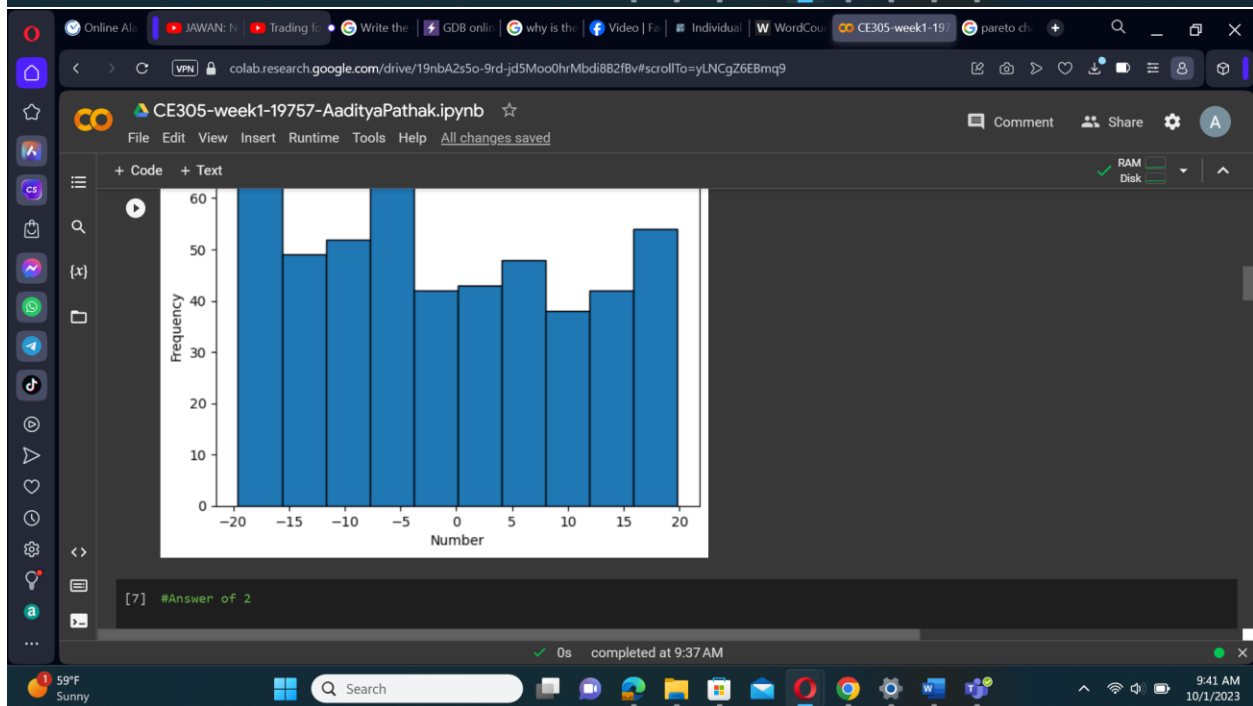
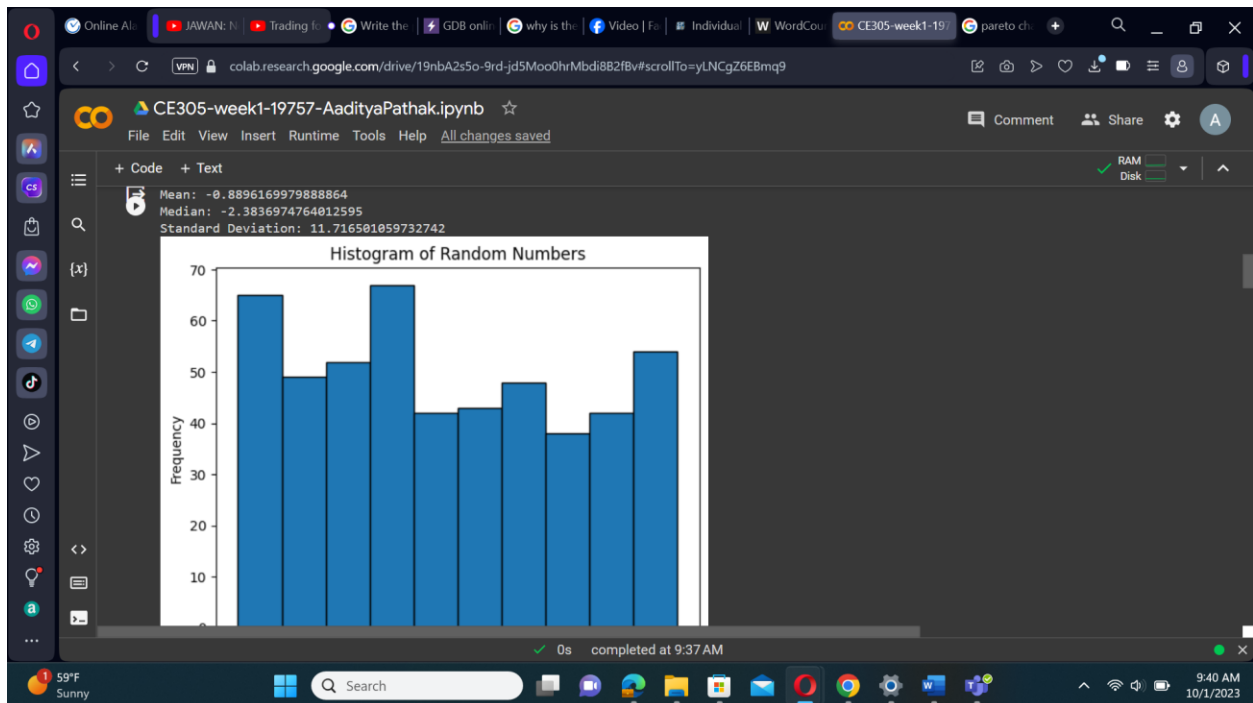
Bottom Screenshot: The notebook is titled "CE305-week1-19757-AadityaPathak.ipynb". The code generates 500 random numbers from -20 to +20. It defines functions to calculate the mean and median. It then calculates the mean and median for the generated numbers and plots a histogram.

```
#Answer of 1
import random
import matplotlib.pyplot as plt

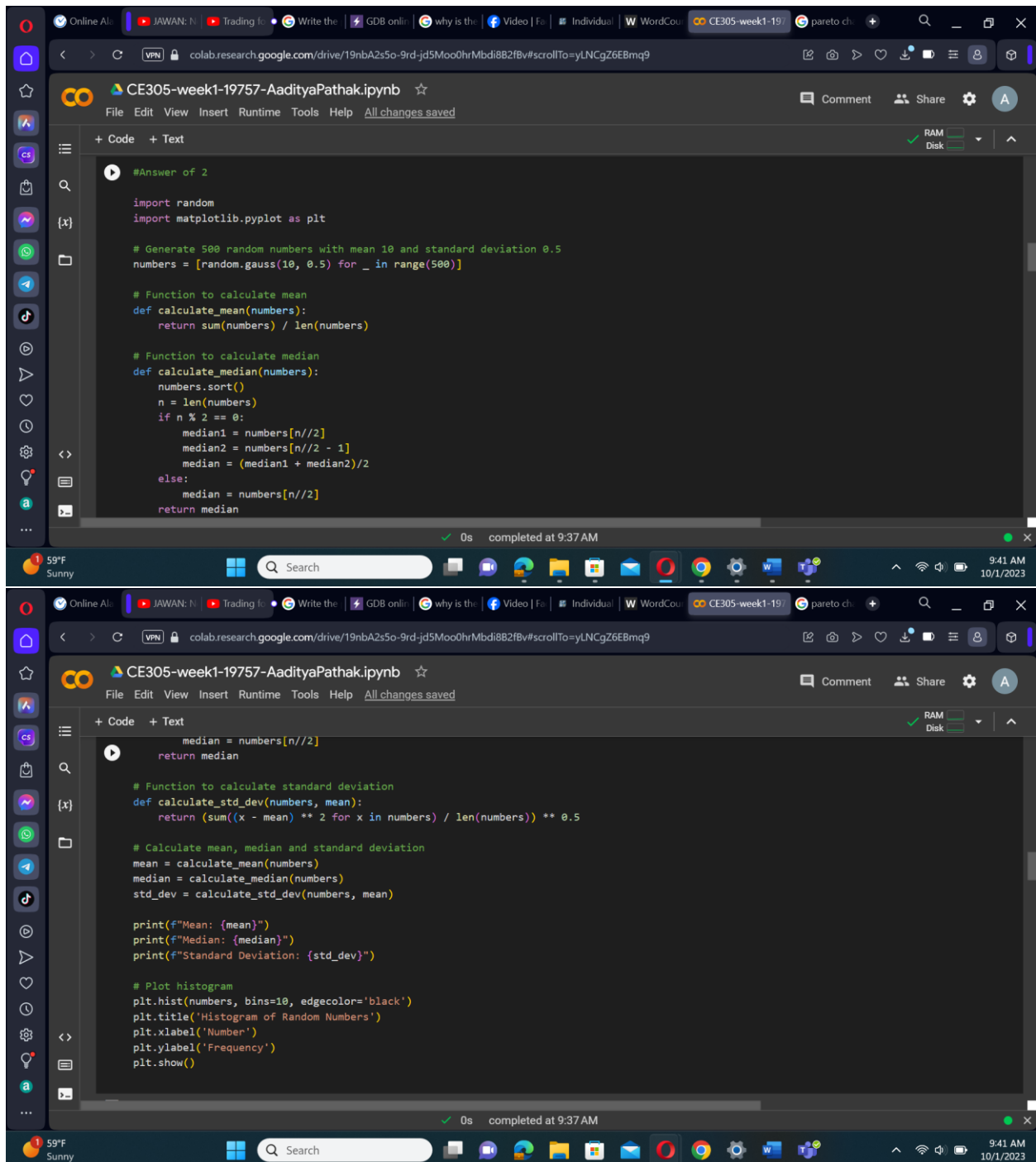
# Generate 500 random numbers from -20 to +20
numbers = [random.uniform(-20, 20) for _ in range(500)]

# Function to calculate mean
def calculate_mean(numbers):
    return sum(numbers) / len(numbers)

# Function to calculate median
def calculate_median(numbers):
    numbers.sort()
    n = len(numbers)
    if n % 2 == 0:
        median1 = numbers[n//2]
        median2 = numbers[n//2 - 1]
        median = (median1 + median2)/2
    else:
        median = numbers[n//2]
    return median
```



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.



The image displays two screenshots of a Google Colab notebook titled "CE305-week1-19757-AadityaPathak.ipynb". The notebook is open in a web browser, showing the code editor and the runtime output.

Top Screenshot: The code defines a function to calculate the mean of a list of numbers. It generates 500 random numbers with a mean of 10 and a standard deviation of 0.5 using `random.gauss(10, 0.5)`. The function `calculate_mean` is defined as follows:

```
#Answer of 2
import random
import matplotlib.pyplot as plt

# Generate 500 random numbers with mean 10 and standard deviation 0.5
numbers = [random.gauss(10, 0.5) for _ in range(500)]

# Function to calculate mean
def calculate_mean(numbers):
    return sum(numbers) / len(numbers)

# Function to calculate median
def calculate_median(numbers):
    numbers.sort()
    n = len(numbers)
    if n % 2 == 0:
        median1 = numbers[n//2]
        median2 = numbers[n//2 - 1]
        median = (median1 + median2)/2
    else:
        median = numbers[n//2]
    return median
```

The notebook shows the execution progress bar at 0s, completed at 9:37 AM.

Bottom Screenshot: The code continues with a function to calculate the standard deviation. It also calculates the mean and median using the previously defined functions. The function `calculate_std_dev` is defined as follows:

```
median = numbers[n//2]
return median

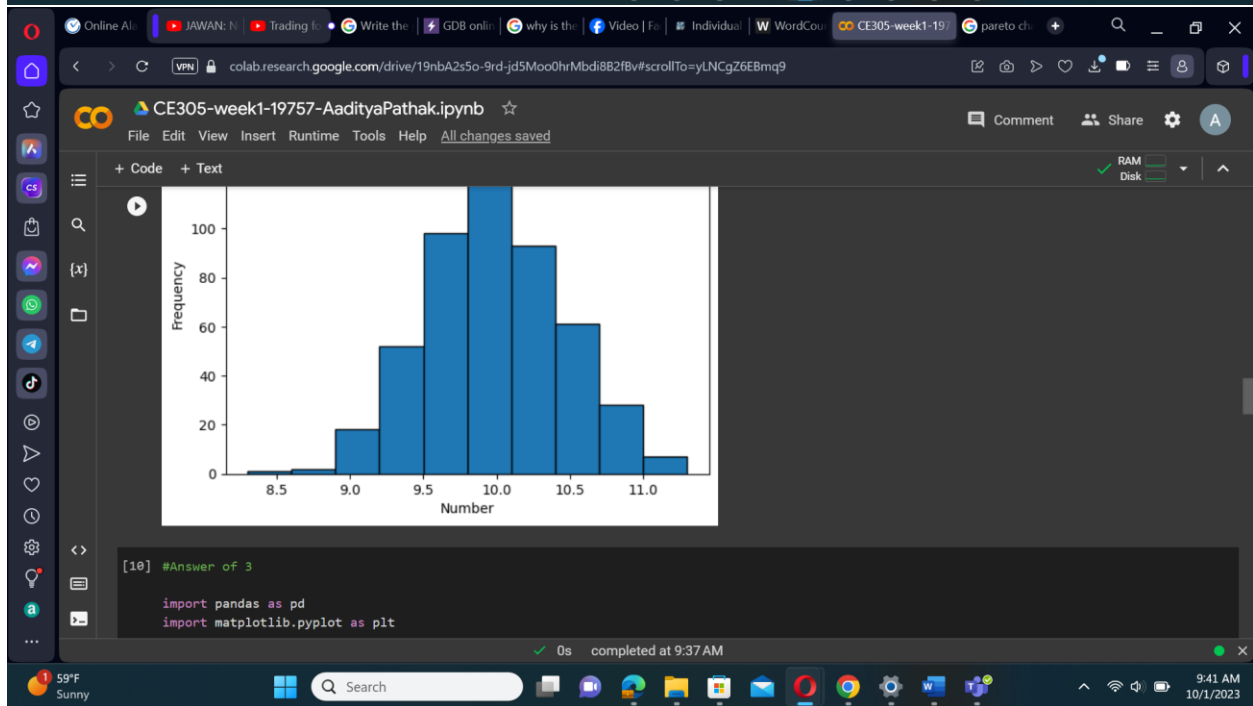
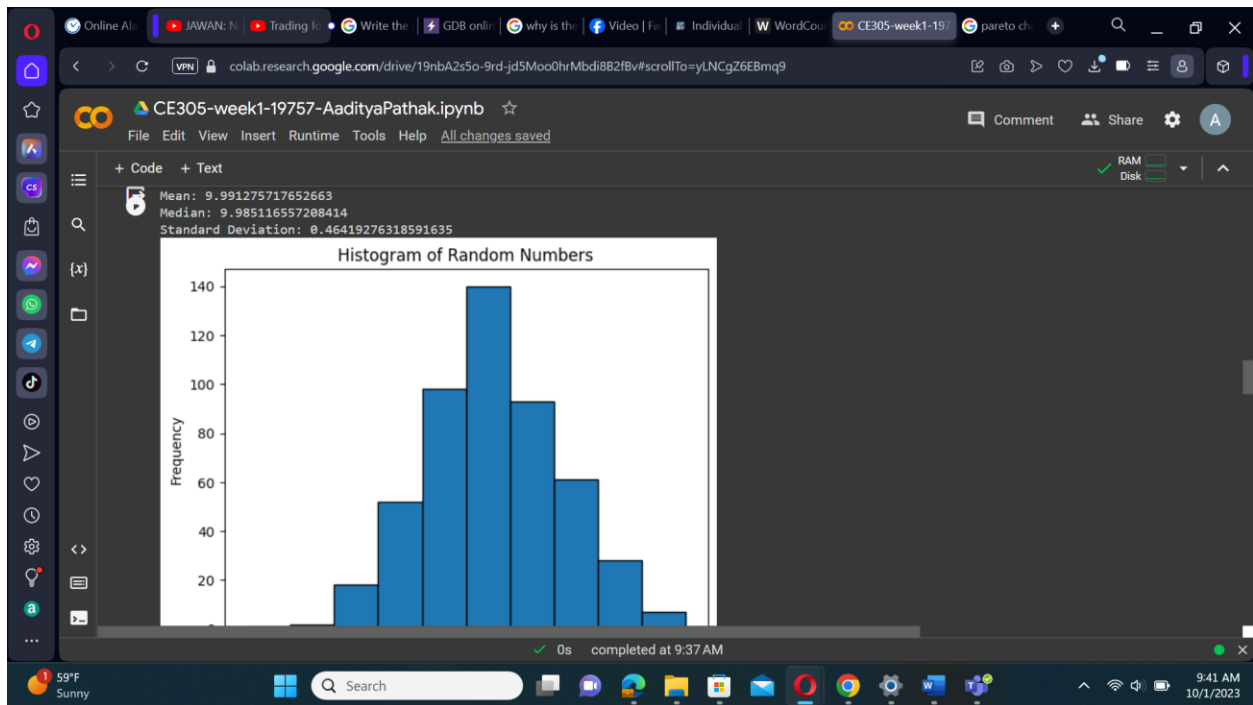
# Function to calculate standard deviation
def calculate_std_dev(numbers, mean):
    return (sum((x - mean) ** 2 for x in numbers) / len(numbers)) ** 0.5

# Calculate mean, median and standard deviation
mean = calculate_mean(numbers)
median = calculate_median(numbers)
std_dev = calculate_std_dev(numbers, mean)

print(f"Mean: {mean}")
print(f"Median: {median}")
print(f"Standard Deviation: {std_dev}")

# Plot histogram
plt.hist(numbers, bins=10, edgecolor='black')
plt.title('Histogram of Random Numbers')
plt.xlabel('Number')
plt.ylabel('Frequency')
plt.show()
```

The notebook shows the execution progress bar at 0s, completed at 9:37 AM.

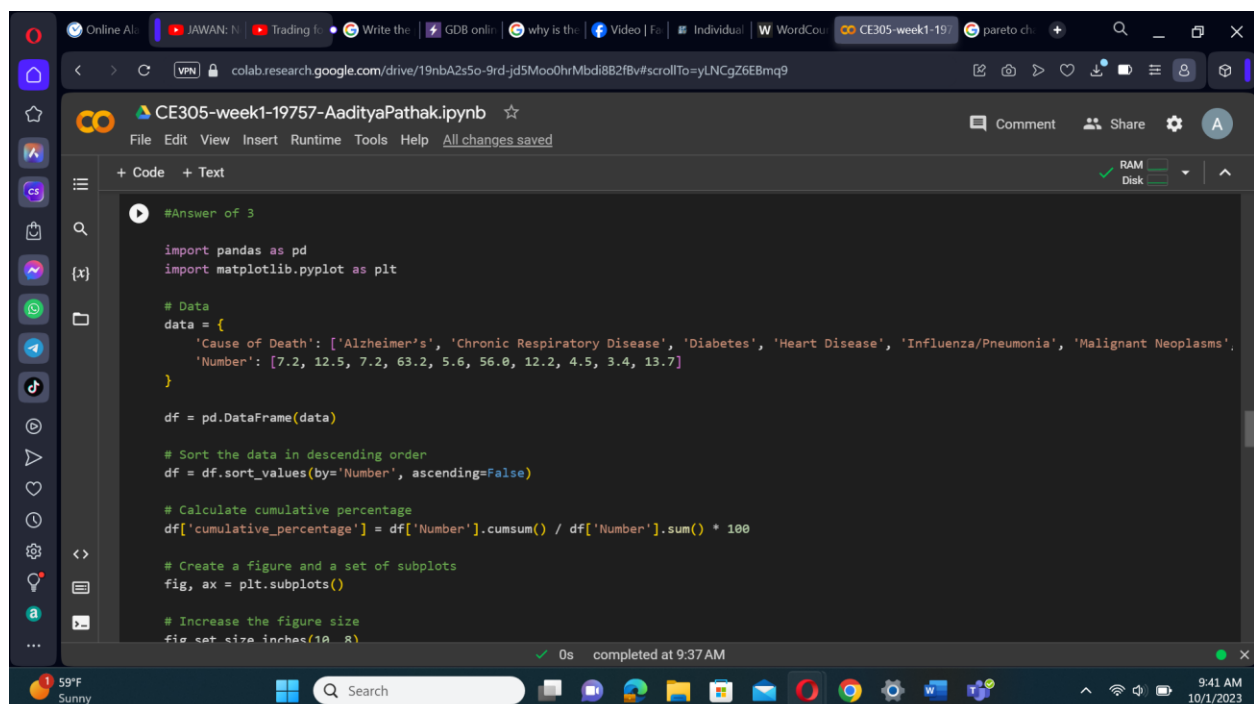


3. The 10-leading causes of death in the United States during 2006 were listed on the Centers for Disease Control and Prevention website. There are a total of 1,855,610 deaths recorded. Plot the Pareto chart in Python or Excel and explain your results.

Cause of Death	Number (x 10,000)
Alzheimer's	7.2
Chronic Respiratory Disease	12.5
Diabetes	7.2
Heart Disease	63.2
Influenza/Pneumonia	5.6
Malignant Neoplasms	56.0
Accidents	12.2
Nephritis/Nephrosis	4.5
Septicemia	3.4
Stroke	13.7

This code first creates a data frame with the given data. It then sorts the data in descending order and calculates the cumulative percentage. The bar plot represents the number of deaths for each cause, and the line plot represents the cumulative percentage. The x-axis labels are the causes of death.

The Pareto chart helps identify the most significant causes of death in terms of their contribution to the total number of deaths. In this case, we can see that Heart Disease and Malignant Neoplasms are the leading causes of death, contributing to a significant portion of the total deaths. The other causes contribute to a lesser extent. This kind of analysis can be useful in prioritizing efforts to prevent these causes of death.



```
#Answer of 3

import pandas as pd
import matplotlib.pyplot as plt

# Data
data = {
    'Cause of Death': ['Alzheimer's', 'Chronic Respiratory Disease', 'Diabetes', 'Heart Disease', 'Influenza/Pneumonia', 'Malignant Neoplasms',
    'Number': [7.2, 12.5, 7.2, 63.2, 5.6, 56.0, 12.2, 4.5, 3.4, 13.7]
}

df = pd.DataFrame(data)

# Sort the data in descending order
df = df.sort_values(by='Number', ascending=False)

# Calculate cumulative percentage
df['cumulative_percentage'] = df['Number'].cumsum() / df['Number'].sum() * 100

# Create a figure and a set of subplots
fig, ax = plt.subplots()

# Increase the figure size
fig.set_size_inches(10, 8)
```

0s completed at 9:37 AM

```
# Increase the figure size
fig.set_size_inches(10, 8)

# Plot bars (Cause of Death)
ax.bar(df['Cause of Death'], df['Number'], color='blue', align='center')

# Plot line (cumulative percentage)
ax2 = ax.twinx()
ax2.plot(df['Cause of Death'], df['cumulative_percentage'], color='red', marker='o', ms=4)

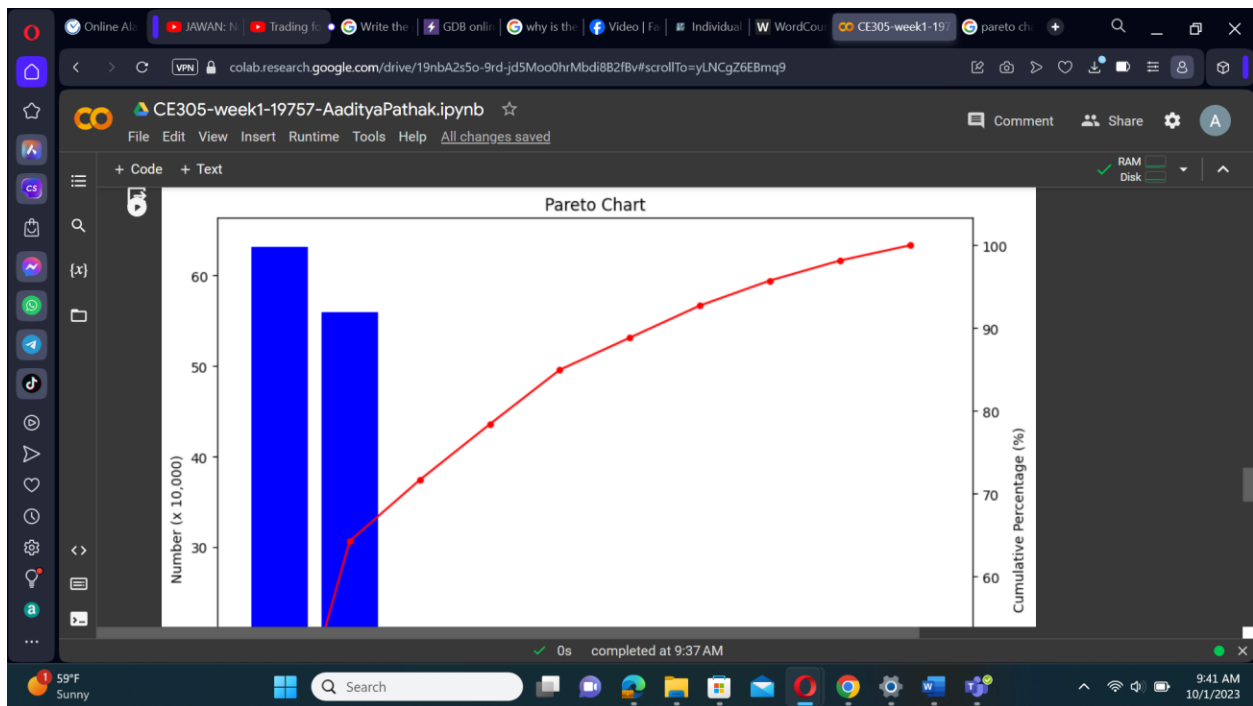
# Add labels and title
plt.title('Pareto Chart')
ax.set_xlabel('Cause of Death')
ax.set_ylabel('Number (x 10,000)')
ax2.set_ylabel('Cumulative Percentage (%)')

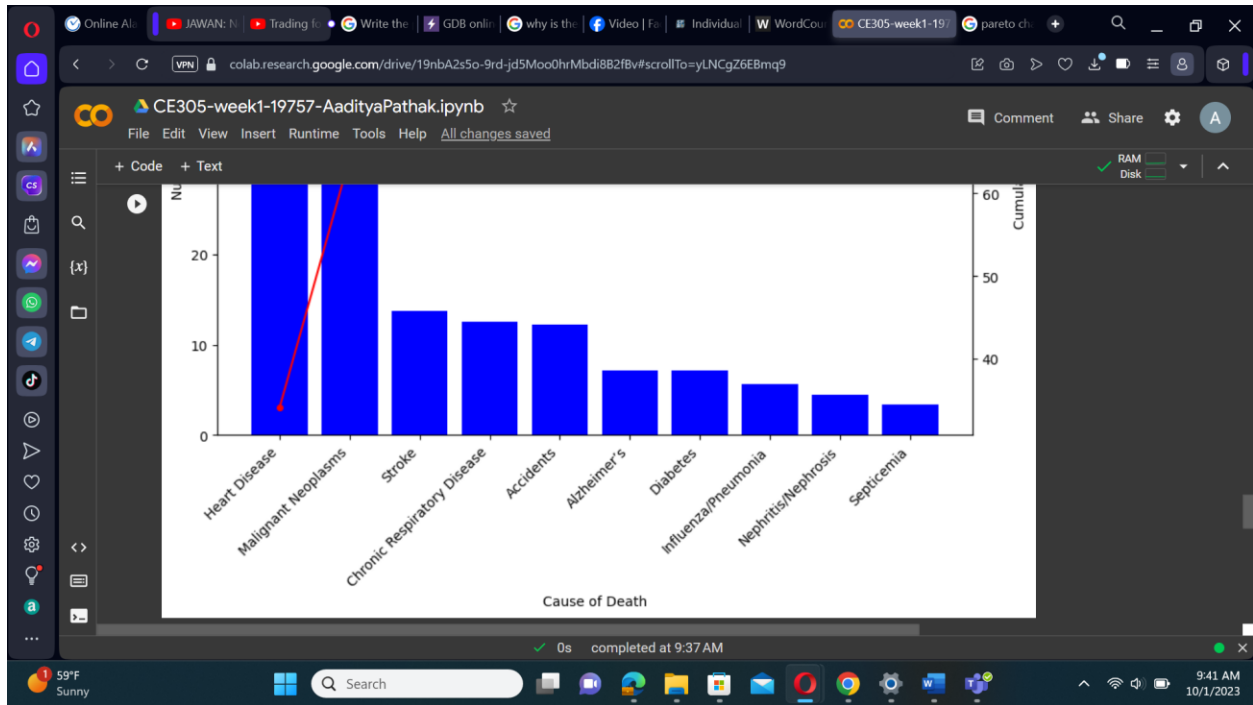
# Rotate x-axis labels for better visibility
plt.setp(ax.get_xticklabels(), rotation=45, horizontalalignment='right')

# Show the plot
plt.show()
```

Pareto Chart

0s completed at 9:37 AM





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.

11	14	15	15	16	16	17	18	19	21	25	36
12	14	15	15	16	16	17	18	19	21	25	39
13	14	15	15	16	17	17	18	20	22	26	43
13	14	15	15	16	17	17	18	20	22	26	46
13	14	15	16	16	17	17	18	20	22	27	50
13	14	15	16	16	17	17	19	20	23	27	54
13	14	15	16	16	17	18	19	20	23	29	59
13	15	15	16	16	17	18	19	20	23	30	67
14	15	15	16	16	17	18	19	21	24	31	
14	15	15	16	16	17	18	19	21	24	34	

The screenshot shows a Google Colab notebook titled "CE305-week1-19757-AadityaPathak.ipynb". The code is as follows:

```
#Answer of 4

import numpy as np
from scipy import stats

# Given data
data = [11, 14, 15, 15, 16, 16, 17, 18, 19, 21, 25, 36,
        12, 14, 15, 15, 16, 16, 17, 18, 19, 21, 25, 39,
        13, 14, 15, 15, 16, 17, 17, 18, 20, 22, 26, 43,
        13, 14, 15, 15, 16, 17, 17, 18, 20, 22, 26, 46,
        13, 14, 15, 16, 16, 17, 17, 18, 20, 22, 27, 50,
        13, 14, 15, 16, 16, 17, 17, 19, 20, 23, 27, 54,
        13, 14, 15, 16, 16, 17, 18, 19, 20, 23, 29, 59,
        13, 15, 15, 16, 16, 17, 18, 19, 20, 23, 30, 67,
        14, 15, 15, 16, 16, 17, 18, 19, 21, 24, 31,
        14, 15, 15, 16, 16, 17, 18, 19, 21, 24, 34]

# Calculate statistics
median = np.median(data)
mode = stats.mode(data).mode
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
p10 = np.percentile(data, 10)
p95 = np.percentile(data, 95)
```

The notebook interface shows the code is executed successfully, with a status bar indicating "0s completed at 9:37 AM". The bottom of the screen shows a Windows taskbar with the date and time "9:41 AM 10/1/2023".

Online A... JAWAN: N Trading fo Write the GDB onlin why is the Video | Fa Individual W WordCou CE305-week1-19/ pareto ch

colab.research.google.com/drive/19nbA2s5o-9rd-jd5Moo0hrMbdI8B2fbv#scrollTo=yLNCgZ6EBmq9

CE305-week1-19757-AadityaPathak.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
# Calculate statistics
median = np.median(data)
mode = stats.mode(data).mode
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
p10 = np.percentile(data, 10)
p95 = np.percentile(data, 95)

# Print results
print(f"Median: {median}")
print(f"Mode: {mode}")
print(f"Q1: {q1}")
print(f"Q3: {q3}")
print(f"P10: {p10}")
print(f"P95: {p95}")
```

Median: 17.0
Mode: 16
Q1: 15.0
Q3: 20.75
P10: 14.0
P95: 39.599999999999966

0s completed at 9:37 AM

59°F Sunny 9:41 AM 10/1/2023