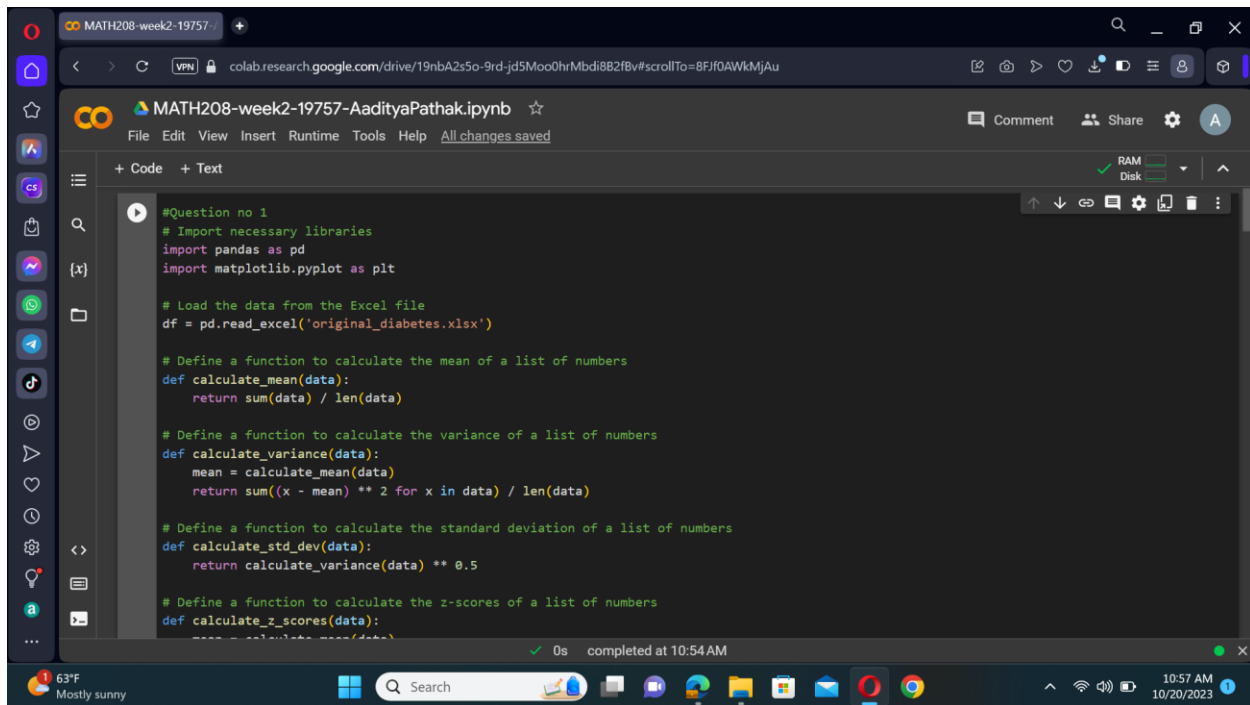


1. Write the program in any computer language to read-in the data from the attached file "original\_diabetes.xlsx" partially coming from Pima Indian Diabetes in the National Institute of Diabetes and Digestive and Kidney Diseases<sup>1</sup>. Find the means, variances, standard deviations, z scores and the values of  $Q_1$ ,  $x$ (median) and  $Q_3$  of "Glucose" and "BloodPressure" by user-defined functions rather than calling existing functions in the libraries. After that, please plot the boxplots of both variables in one frame.



```
#Question no 1
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt

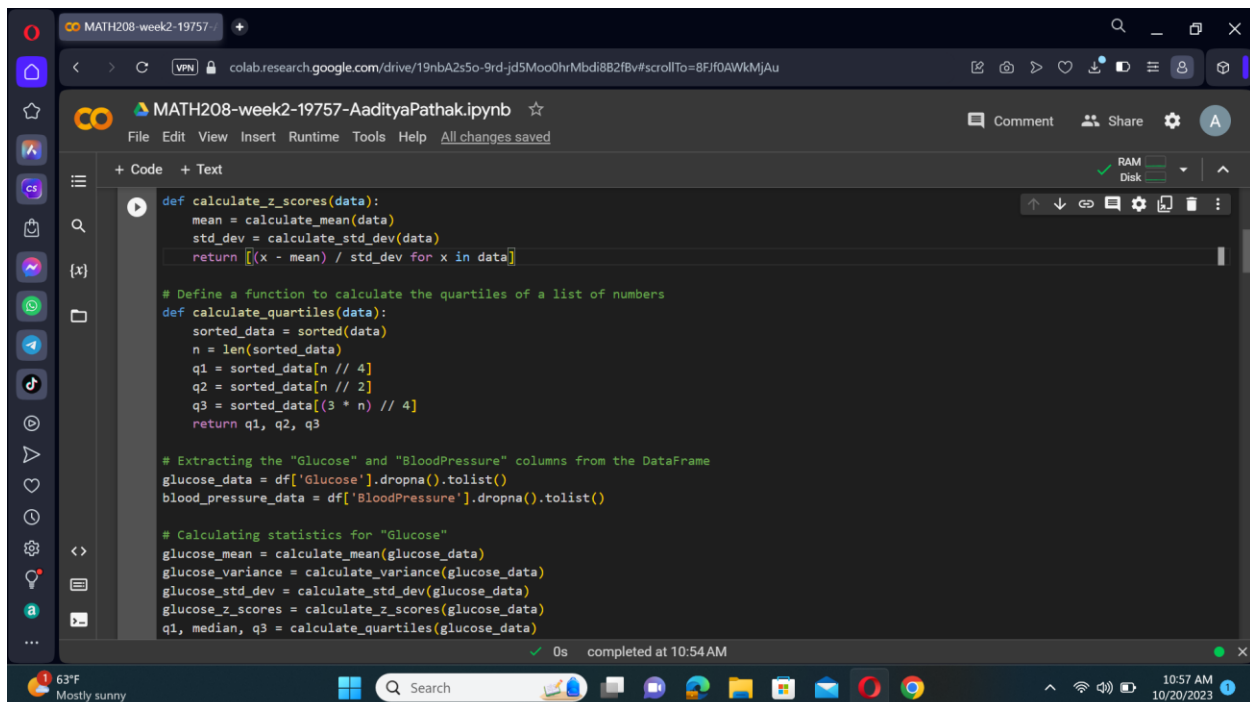
# Load the data from the Excel file
df = pd.read_excel('original_diabetes.xlsx')

# Define a function to calculate the mean of a list of numbers
def calculate_mean(data):
    return sum(data) / len(data)

# Define a function to calculate the variance of a list of numbers
def calculate_variance(data):
    mean = calculate_mean(data)
    return sum((x - mean) ** 2 for x in data) / len(data)

# Define a function to calculate the standard deviation of a list of numbers
def calculate_std_dev(data):
    return calculate_variance(data) ** 0.5

# Define a function to calculate the z-scores of a list of numbers
def calculate_z_scores(data):
```



```
def calculate_z_scores(data):
    mean = calculate_mean(data)
    std_dev = calculate_std_dev(data)
    return [(x - mean) / std_dev for x in data]

# Define a function to calculate the quartiles of a list of numbers
def calculate_quartiles(data):
    sorted_data = sorted(data)
    n = len(sorted_data)
    q1 = sorted_data[n // 4]
    q2 = sorted_data[n // 2]
    q3 = sorted_data[(3 * n) // 4]
    return q1, q2, q3

# Extracting the "Glucose" and "BloodPressure" columns from the DataFrame
glucose_data = df['Glucose'].dropna().tolist()
blood_pressure_data = df['BloodPressure'].dropna().tolist()

# Calculating statistics for "Glucose"
glucose_mean = calculate_mean(glucose_data)
glucose_variance = calculate_variance(glucose_data)
glucose_std_dev = calculate_std_dev(glucose_data)
glucose_z_scores = calculate_z_scores(glucose_data)
q1, median, q3 = calculate_quartiles(glucose_data)
```

```
MATH208-week2-19757-
colab.research.google.com/drive/19nbA2s5o-9rd-jd5Moo0hrMbdI8B2fbv#scrollTo=8FJf0AWkMjAu

MATH208-week2-19757-AadityaPathak.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text
q1, median, q3 = calculate_quartiles(glucose_data)

# Calculating statistics for "BloodPressure"
bp_mean = calculate_mean(blood_pressure_data)
bp_variance = calculate_variance(blood_pressure_data)
bp_std_dev = calculate_std_dev(blood_pressure_data)
bp_z_scores = calculate_z_scores(blood_pressure_data)
bp_q1, bp_median, bp_q3 = calculate_quartiles(blood_pressure_data)

# Print the calculated statistics for "Glucose"
print("Statistics for Glucose:")
print(f"Mean: {glucose_mean}")
print(f"Variance: {glucose_variance}")
print(f"Standard Deviation: {glucose_std_dev}")
print(f"Z-Scores: {glucose_z_scores}")
print(f"Q1: {q1}, Median (Q2): {median}, Q3: {q3}")
print()

# Print the calculated statistics for "BloodPressure"
print("Statistics for BloodPressure:")
print(f"Mean: {bp_mean}")
print(f"Variance: {bp_variance}")
print(f"Standard Deviation: {bp_std_dev}")
print(f"Z-Scores: {bp_z_scores}")

0s completed at 10:54 AM
63°F Mostly sunny 10:57 AM 10/20/2023
```

```
MATH208-week2-19757-
colab.research.google.com/drive/19nbA2s5o-9rd-jd5Moo0hrMbdI8B2fbv#scrollTo=8FJf0AWkMjAu

MATH208-week2-19757-AadityaPathak.ipynb
File Edit View Insert Runtime Tools Help All changes saved

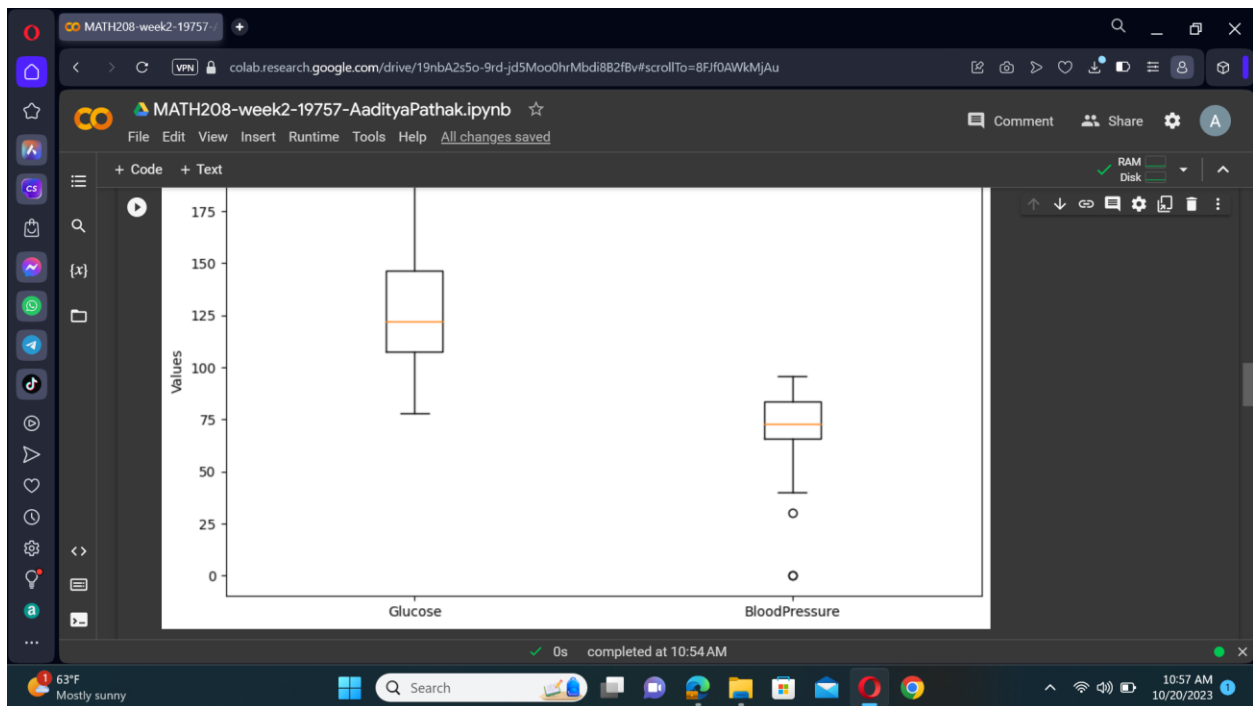
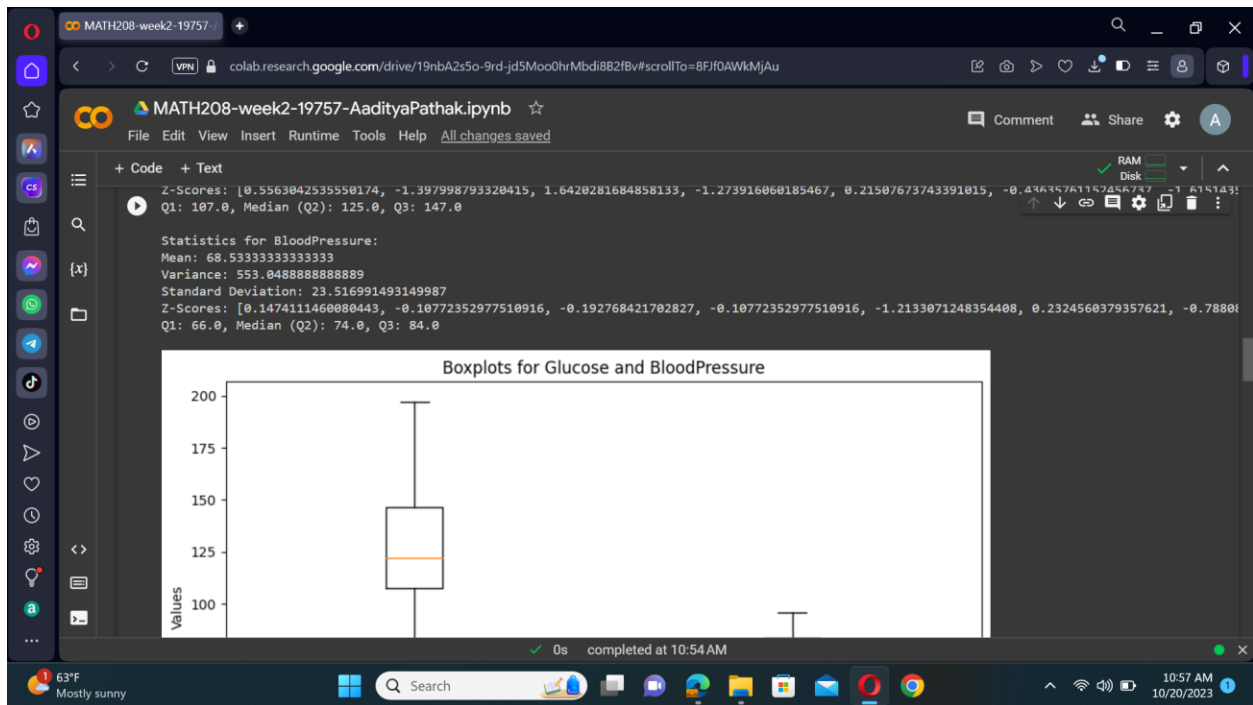
+ Code + Text
print(f"Variance: {bp_variance}")
print(f"Standard Deviation: {bp_std_dev}")
print(f"Z-Scores: {bp_z_scores}")
print(f"Q1: {bp_q1}, Median (Q2): {bp_median}, Q3: {bp_q3}")
print()

# Plotting boxplots for "Glucose" and "BloodPressure" in one frame
plt.figure(figsize=(10, 6))
plt.boxplot([glucose_data, blood_pressure_data], labels=["Glucose", "BloodPressure"])
plt.title("Boxplots for Glucose and BloodPressure")
plt.ylabel("Values")
plt.show()

Statistics for Glucose:
Mean: 130.06666666666666
Variance: 1039.1955555555553
Standard Deviation: 32.23655619875602
Z-Scores: [0.5563842535550174, -1.397998793320415, 1.6420281684858133, -1.273916060185467, 0.21507673743391015, -0.43635761152456737, -1.615143]
Q1: 107.0, Median (Q2): 125.0, Q3: 147.0

Statistics for BloodPressure:
Mean: 68.53333333333333
Variance: 553.0488888888889
Standard Deviation: 23.516991493149987

0s completed at 10:54 AM
63°F Mostly sunny 10:57 AM 10/20/2023
```



2. Write the program to verify Chebyshev's inequality as follows by 50 random numbers generated from a normal distribution with mean  $\mu = 10$  and standard deviation  $\sigma = 0.5$ .

$$P(-k\sigma < X - \mu < k\sigma) \geq (1 - 1/k^2) \text{ or } P(|X - \mu| \geq k\sigma) \leq 1/k^2$$

For instance,

$$\text{if } k = 1, P(-\sigma < X - \mu < \sigma) = P(\mu - \sigma < X < \mu + \sigma)$$

= how many random numbers are within the range  $[\mu - \sigma, \mu + \sigma]$

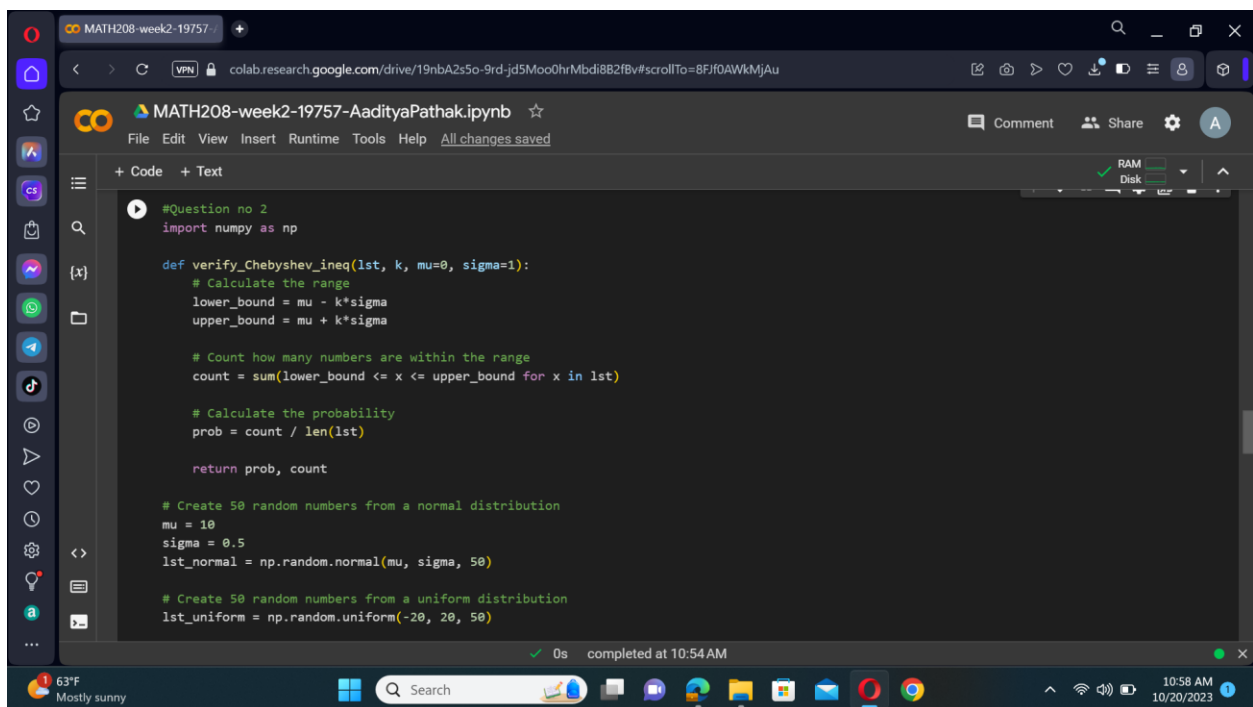
Total random numbers

$$\text{if } k = 2, P(-2\sigma < X - \mu < 2\sigma) = P(\mu - 2\sigma < X < \mu + 2\sigma)$$

= how many random numbers are within the range  $[\mu - 2\sigma, \mu + 2\sigma]$

Total random numbers

Repeat the similar process to verify Chebyshev's inequality as well by 50 random numbers within the range  $[-20, +20]$  generated from a uniform distribution.



```
#Question no 2
import numpy as np

def verify_Chebyshev_ineq(lst, k, mu=0, sigma=1):
    # Calculate the range
    lower_bound = mu - k*sigma
    upper_bound = mu + k*sigma

    # Count how many numbers are within the range
    count = sum(lower_bound <= x <= upper_bound for x in lst)

    # Calculate the probability
    prob = count / len(lst)

    return prob, count

# Create 50 random numbers from a normal distribution
mu = 10
sigma = 0.5
lst_normal = np.random.normal(mu, sigma, 50)

# Create 50 random numbers from a uniform distribution
lst_uniform = np.random.uniform(-20, 20, 50)
```

The screenshot shows a Google Colab notebook interface. The top bar indicates the notebook is named 'MATH208-week2-19757-AadityaPathak.ipynb'. The code editor displays a Python function 'verify\_Chebyshev\_ineq' that takes a list 'lst', a parameter 'k', and default values for 'mu' (0) and 'sigma' (1). The function calculates the range [mu - k\*sigma, mu + k\*sigma], counts the number of elements in 'lst' that fall within this range, and returns the probability (count / len(lst)) and the count. Below the function, two lists are generated: 'lst\_normal' from a normal distribution with mu=10 and sigma=0.5, and 'lst\_uniform' from a uniform distribution between -20 and 20. The bottom status bar shows '0s completed at 10:54 AM'.

MATH208-week2-19757- AadityaPathak.ipynb

```
# Testcases
for lst in [lst_normal, lst_uniform]:
    for k in [1, 2**0.5, 1.5, 2, 3]:
        prob, count = verify_Chebyshev_ineq(lst, k, mu, sigma)
        print(f'When k = {k}, Probability of |X-u| = {prob}; 1-1/(k^2)= {1-1/k**2}')
        if prob >= 1-1/k**2:
            print(f'When k = {k} , P(|X-u|<k*sd)>=1-1/k^2 is True')
        else:
            print(f'When k = {k} , P(|X-u|<k*sd)>=1-1/k^2 is False')
```

When k = 1, Probability of |X-u| = 0.62 ; 1-1/(k^2)= 0.0  
When k = 1 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 1.4142135623730951, Probability of |X-u| = 0.76 ; 1-1/(k^2)= 0.5000000000000001  
When k = 1.4142135623730951 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 1.5, Probability of |X-u| = 0.76 ; 1-1/(k^2)= 0.5555555555555556  
When k = 1.5 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 2, Probability of |X-u| = 0.94 ; 1-1/(k^2)= 0.75  
When k = 2 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 3, Probability of |X-u| = 0.98 ; 1-1/(k^2)= 0.8888888888888888  
When k = 3 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 1, Probability of |X-u| = 0.02 ; 1-1/(k^2)= 0.0  
When k = 1 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 1.4142135623730951, Probability of |X-u| = 0.06 ; 1-1/(k^2)= 0.5000000000000001  
When k = 1.4142135623730951 , P(|X-u|<k\*sd)>=1-1/k^2 is False

0s completed at 10:54 AM

MATH208-week2-19757- AadityaPathak.ipynb

```
print(f'When k = {k} , P(|X-u|<k*sd)>=1-1/k^2 is True')
else:
    print(f'When k = {k} , P(|X-u|<k*sd)>=1-1/k^2 is False')
```

When k = 1, Probability of |X-u| = 0.62 ; 1-1/(k^2)= 0.0  
When k = 1 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 1.4142135623730951, Probability of |X-u| = 0.76 ; 1-1/(k^2)= 0.5000000000000001  
When k = 1.4142135623730951 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 1.5, Probability of |X-u| = 0.76 ; 1-1/(k^2)= 0.5555555555555556  
When k = 1.5 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 2, Probability of |X-u| = 0.94 ; 1-1/(k^2)= 0.75  
When k = 2 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 3, Probability of |X-u| = 0.98 ; 1-1/(k^2)= 0.8888888888888888  
When k = 3 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 1, Probability of |X-u| = 0.02 ; 1-1/(k^2)= 0.0  
When k = 1 , P(|X-u|<k\*sd)>=1-1/k^2 is True  
When k = 1.4142135623730951, Probability of |X-u| = 0.06 ; 1-1/(k^2)= 0.5000000000000001  
When k = 1.4142135623730951 , P(|X-u|<k\*sd)>=1-1/k^2 is False  
When k = 1.5, Probability of |X-u| = 0.06 ; 1-1/(k^2)= 0.5555555555555556  
When k = 1.5 , P(|X-u|<k\*sd)>=1-1/k^2 is False  
When k = 2, Probability of |X-u| = 0.08 ; 1-1/(k^2)= 0.75  
When k = 2 , P(|X-u|<k\*sd)>=1-1/k^2 is False  
When k = 3, Probability of |X-u| = 0.14 ; 1-1/(k^2)= 0.8888888888888888  
When k = 3 , P(|X-u|<k\*sd)>=1-1/k^2 is False

0s completed at 10:54 AM

3. Given the following dataset, write the program to fit it by linear regression showing the values of  $b_1$ ,  $b_0$  and coefficient of linear correlation  $r$ . After that, please plot the curve of  $X$  vs  $Y$  and straight fitting line. Can we draw the conclusion that linear model is good for the dataset if the value of  $r$  is very close to  $+1$ ? Suggest which fitting model should be better than linear based on the data visualization of the given dataset.

$X$   $Y$

2 30

3 25

4 95

5 115

6 265

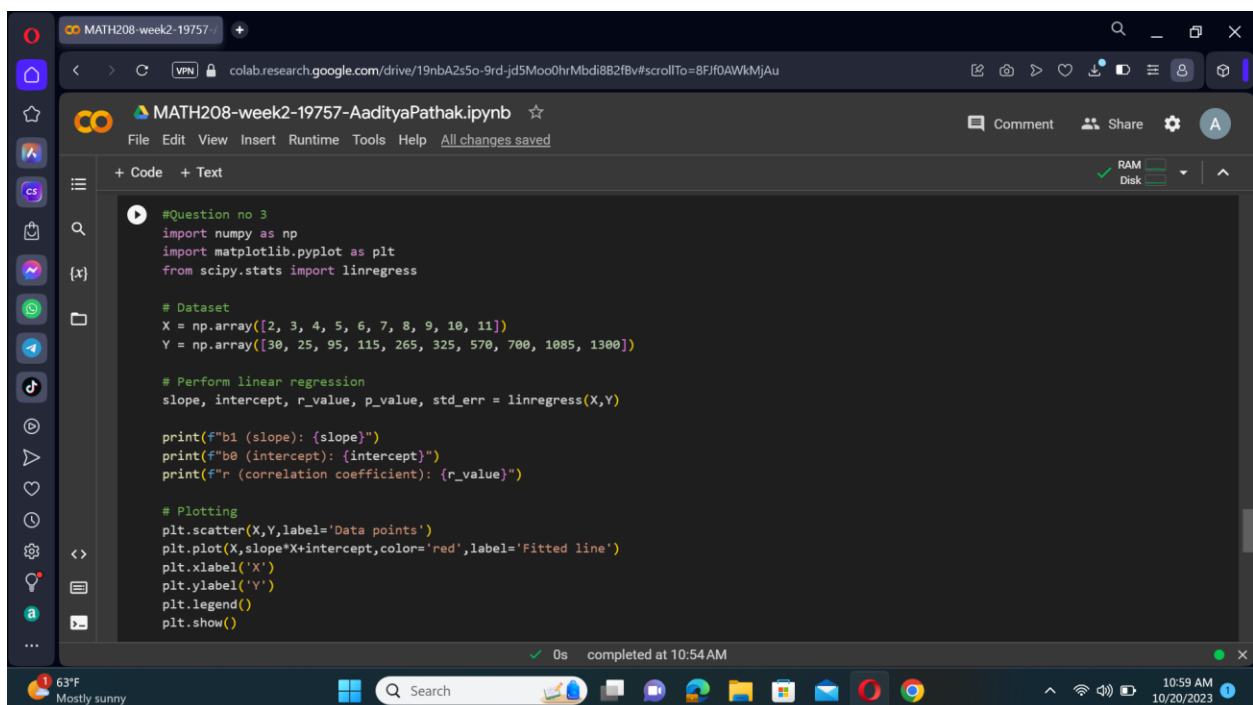
7 325

8 570

9 700

10 1085

11 1300



The screenshot shows a Google Colab notebook titled "MATH208-week2-19757-AadityaPathak.ipynb". The code in the notebook is as follows:

```
#Question no 3
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress

# Dataset
X = np.array([2, 3, 4, 5, 6, 7, 8, 9, 10, 11])
Y = np.array([30, 25, 95, 115, 265, 325, 570, 700, 1085, 1300])

# Perform linear regression
slope, intercept, r_value, p_value, std_err = linregress(X,Y)

print(f"b1 (slope): {slope}")
print(f"b0 (intercept): {intercept}")
print(f"r (correlation coefficient): {r_value}")

# Plotting
plt.scatter(X,Y,label='Data points')
plt.plot(X,slope*X+intercept,color='red',label='Fitted line')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```

The output of the code is:

```
b1 (slope): 125.5
b0 (intercept): -245.5
r (correlation coefficient): 0.9999999999999999
```

The notebook interface shows the code is completed at 10:54 AM. The bottom status bar indicates the system temperature is 63°F and mostly sunny, with the time 10:59 AM on 10/20/2023.

