San Francisco Bay University

MATH208 - Probability and Statistics
2023 Fall Homework #2

Due day: 10/16/2023

Instruction:

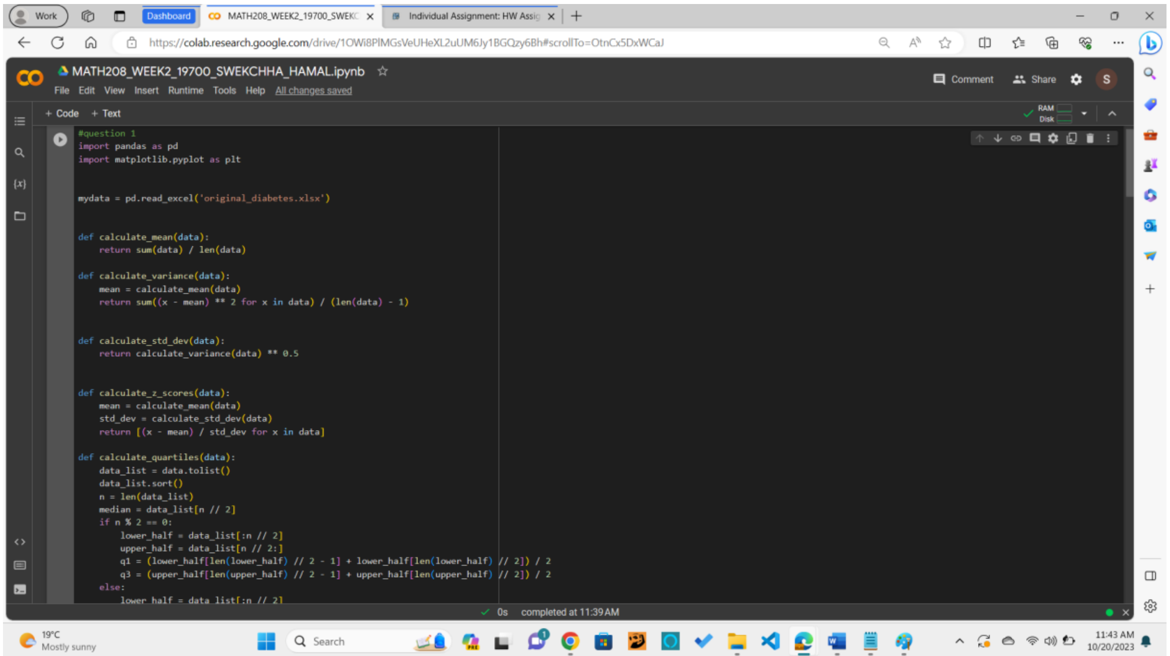
1. Homework answer sheet should contain the original questions and corresponding answers.
2. Answer sheet must be in PDF file format with Github links for the programming questions, but MS Word file can’t be accepted. As follows is the answer sheet name format.
<course_id>_week<week_number>_StudentID_FirstName_LastName.pdf
3. The program name in Github must follow the format like
<course_id>_week<week_number>_q<question_number>_StudentID_FirstName_LastName
4. If the calculation in Excel is needed, the original file must be provided.
5. Show screenshot of all running results, including the system date/time.
6. The calculation process must be printed if needed, handwriting can’t be accepted.
7. Only accept homework submission uploaded via Canvas.
8. Overdue homework submission can’t be accepted.
3. Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)

For the students in Engineering School

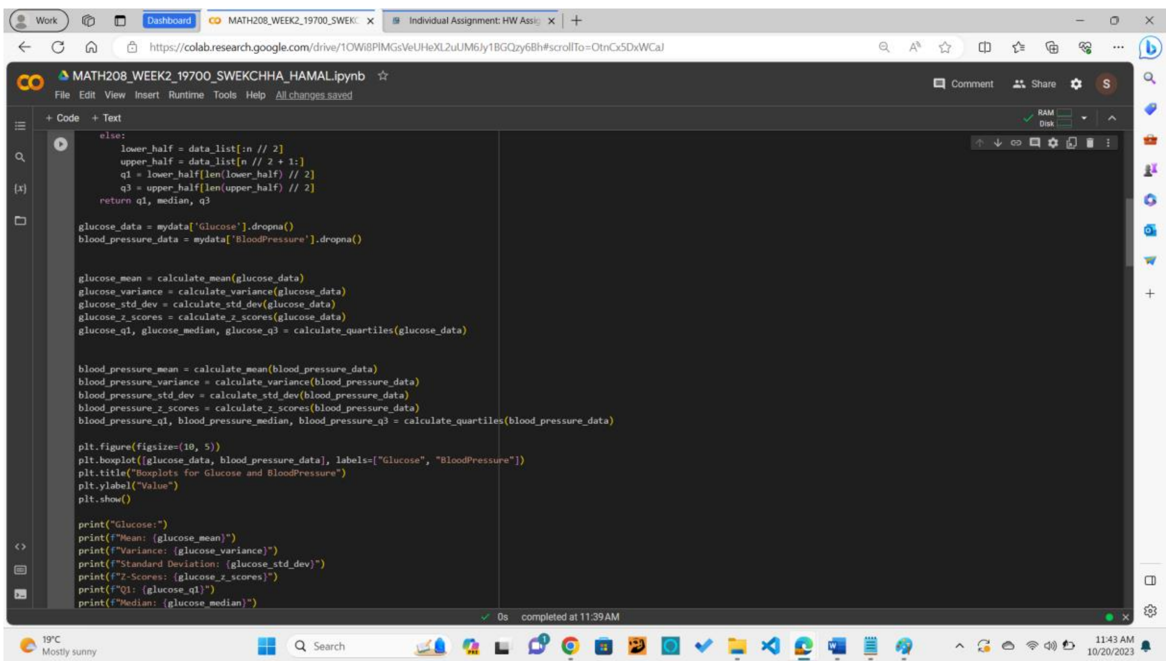
1. Write the program in any computer language to read-in the data from the attached file "*original_diabetes.xlsx*" partially coming from Pima Indian Diabetes in the National Institute of Diabetes and Digestive and Kidney Diseases[1]. Find the means, variances, standard deviations, z scores and the values of Q_1, $\tilde{x}$(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.

[1] https://github.com/npradaschnor/Pima-Indians-Diabetes-Dataset

Ans:



```
#question 1
import pandas as pd
import matplotlib.pyplot as plt


mydata = pd.read_excel('original_diabetes.xlsx')


def calculate_mean(data):
    return sum(data) / len(data)

def calculate_variance(data):
    mean = calculate_mean(data)
    return sum((x - mean) ** 2 for x in data) / (len(data) - 1)


def calculate_std_dev(data):
    return calculate_variance(data) ** 0.5


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

def calculate_quartiles(data):
    data_list = data.tolist()
    data_list.sort()
    n = len(data_list)
    median = data_list[n // 2]
    if n % 2 == 0:
        lower_half = data_list[:n // 2]
        upper_half = data_list[n // 2:]
        q1 = (lower_half[len(lower_half) // 2 - 1] + lower_half[len(lower_half) // 2]) / 2
        q3 = (upper_half[len(upper_half) // 2 - 1] + upper_half[len(upper_half) // 2]) / 2
    else:
        lower_half = data_list[:n // 2]
```



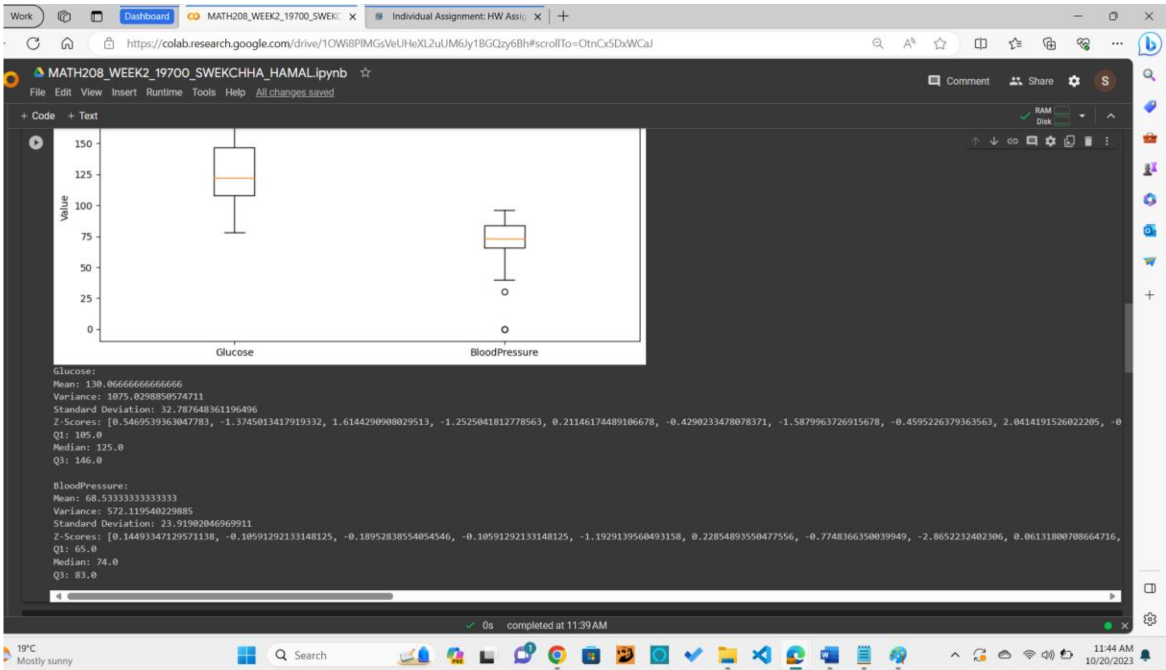
```
    else:
        lower_half = data_list[:n // 2]
        upper_half = data_list[n // 2 + 1:]
        q1 = lower_half[len(lower_half) // 2]
        q3 = upper_half[len(upper_half) // 2]
    return q1, median, q3

glucose_data = mydata['Glucose'].dropna()
blood_pressure_data = mydata['BloodPressure'].dropna()


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)
glucose_q1, glucose_median, glucose_q3 = calculate_quartiles(glucose_data)


blood_pressure_mean = calculate_mean(blood_pressure_data)
blood_pressure_variance = calculate_variance(blood_pressure_data)
blood_pressure_std_dev = calculate_std_dev(blood_pressure_data)
blood_pressure_z_scores = calculate_z_scores(blood_pressure_data)
blood_pressure_q1, blood_pressure_median, blood_pressure_q3 = calculate_quartiles(blood_pressure_data)

plt.figure(figsize=(10, 5))
plt.boxplot([glucose_data, blood_pressure_data], labels=["Glucose", "BloodPressure"])
plt.title("Boxplots for Glucose and BloodPressure")
plt.ylabel("Value")
plt.show()

print("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: {glucose_q1}")
print(f"Median: {glucose_median}")
```

MATH208_WEEK2_19700_SWEKCHHA_HAMAL.ipynb

```python
glucose_std_dev = calculate_std_dev(glucose_data)
glucose_z_scores = calculate_z_scores(glucose_data)
glucose_q1, glucose_median, glucose_q3 = calculate_quartiles(glucose_data)


blood_pressure_mean = calculate_mean(blood_pressure_data)
blood_pressure_variance = calculate_variance(blood_pressure_data)
blood_pressure_std_dev = calculate_std_dev(blood_pressure_data)
blood_pressure_z_scores = calculate_z_scores(blood_pressure_data)
blood_pressure_q1, blood_pressure_median, blood_pressure_q3 = calculate_quartiles(blood_pressure_data)

plt.figure(figsize=(10, 5))
plt.boxplot([glucose_data, blood_pressure_data], labels=["Glucose", "BloodPressure"])
plt.title("Boxplots for Glucose and BloodPressure")
plt.ylabel("Value")
plt.show()

print("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: {glucose_q1}")
print(f"Median: {glucose_median}")
print(f"Q3: {glucose_q3}")


print("\nBloodPressure:")
print(f"Mean: {blood_pressure_mean}")
print(f"Variance: {blood_pressure_variance}")
print(f"Standard Deviation: {blood_pressure_std_dev}")
print(f"Z-Scores: {blood_pressure_z_scores}")
print(f"Q1: {blood_pressure_q1}")
print(f"Median: {blood_presure_median}")
print(f"Q3: {blood_pressure_q3}")
```

MATH208_WEEK2_19700_SWEKCHHA_HAMAL.ipynb


```
Glucose:
Mean: 130.06666666666666
Variance: 1075.0298850574711
Standard Deviation: 32.787648361196496
Z-Scores: [0.5469539363047783, -1.3745013417919332, 1.6144290908029513, -1.2525041812778563, 0.21146174489106678, -0.4290233478078371, -1.5879963726915678, -0.4595226379363563, 2.041419152602205, -0
Q1: 105.0
Median: 125.0
Q3: 146.0

BloodPressure:
Mean: 68.53333333333333
Variance: 572.119540229885
Standard Deviation: 23.91902046969911
```



2. Write the program to verify Chebyshev's inequity as follows by *50* random numbers generated from a normal distribution with mean μ = *10* and standard deviation σ = *0.5*.

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

For instance,

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

$$= \frac{\text{how many random numbers are within the range } [\mu-\sigma, \mu+\sigma]}{\text{Total random numbers}}$$

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

$$= \frac{\text{how many random numbers are within the range } [\mu-2\sigma, \mu+2\sigma]}{\text{Total random numbers}}$$

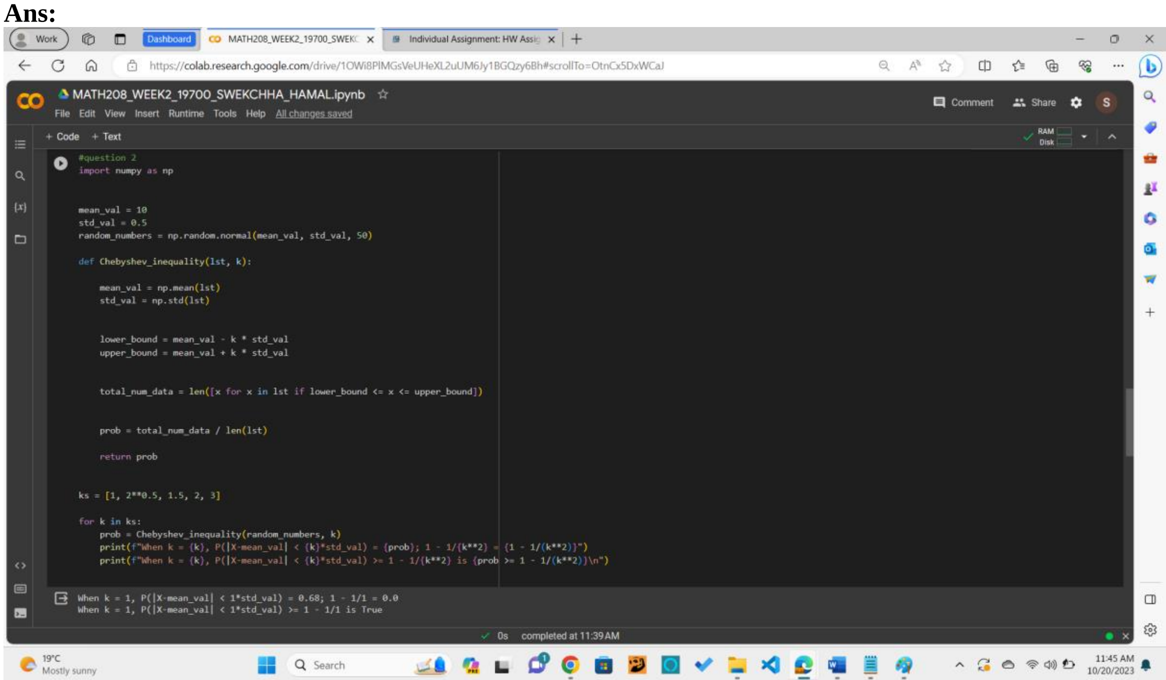
Repeat the similar process to verify Chebyshev's inequity as well by *50* random numbers within the range [-20, +20] generated from a uniform distribution.

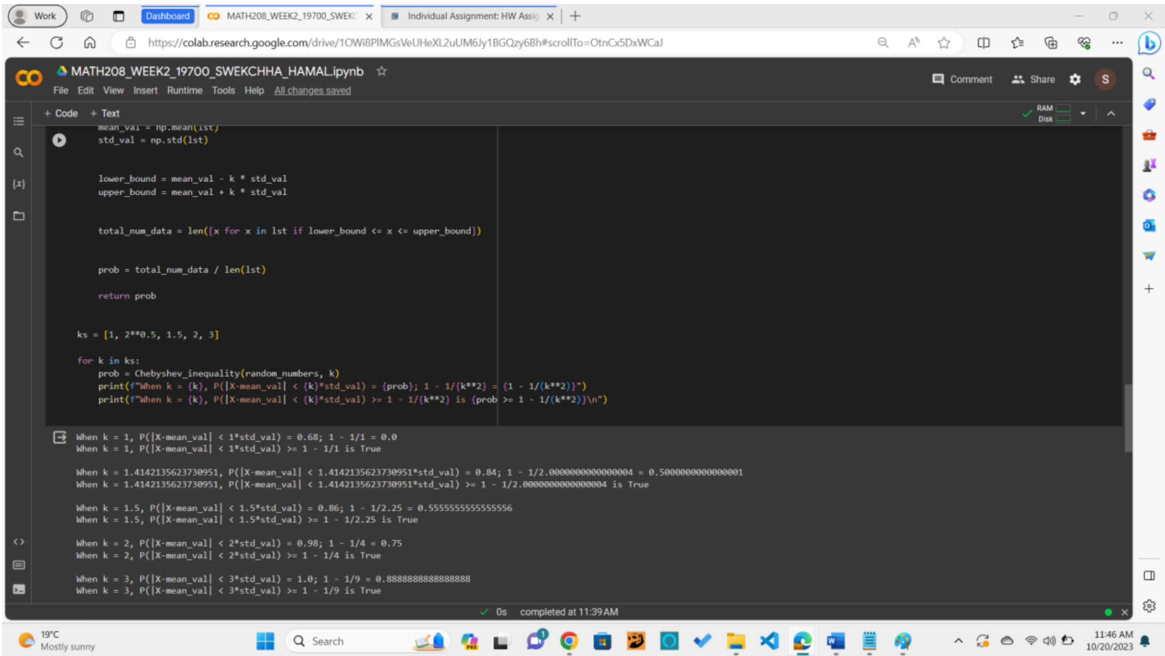
As below are the program structure and testcases.

```
"""
   Create 50 random numbers here first.
   … …

   def   verify_Chebyshev_ineq(lst, k):            # lst: random numbers in list dType
        # your program is here
        … …
        # Return how many numbers are within the range [μ − kσ, μ + kσ]

   # Testcases
   k=1
```

```
cnt=verify_Chebyshev_ineq(lst, k)
#---- running results for example --------
Probability of |X-u| = 0.64 ; 1-1/(k^2)= 0.0
When k = 1 , P(|X-u|<k*sd)>=1-1/k^2 is True

k=2**0.5                    # k=√2
cnt=verify_Chebyshev_ineq(lst, k)
#---- running results for example --------
Probability of |X-u| = 0.84 ; 1-1/(k^2)= 0.5000000000000001
When k = 1.4142135623730951 , P(|X-u|<k*sd)>=1-1/k^2 is True

k=1.5
cnt=verify_Chebyshev_ineq(lst, k)
#---- running results for example --------
Probability of |X-u| = 0.9 ; 1-1/(k^2)= 0.5555555555555556
When k = 1.5 , P(|X-u|<k*sd)>=1-1/k^2 is True

k=2
cnt=verify_Chebyshev_ineq(lst, k)
#---- running results for example --------
Probability of |X-u| = 1.0 ; 1-1/(k^2)= 0.75
When k = 2 , P(|X-u|<k*sd)>=1-1/k^2 is True

k=3
cnt=verify_Chebyshev_ineq(lst, k)
#---- running results for example --------
Probability of |X-u| = 1.0 ; 1-1/(k^2)= 0.8888888888888888
When k = 3 ,P(|X-u|<k*sd)>=1-1/k^2 is True


"""
```

Ans:

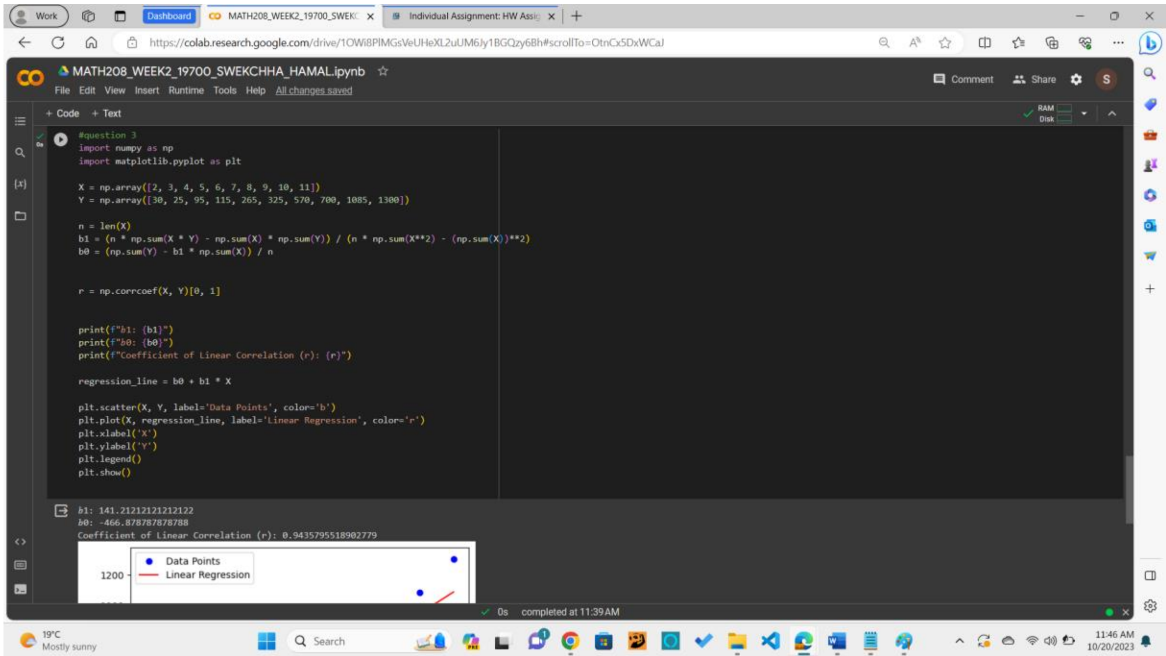


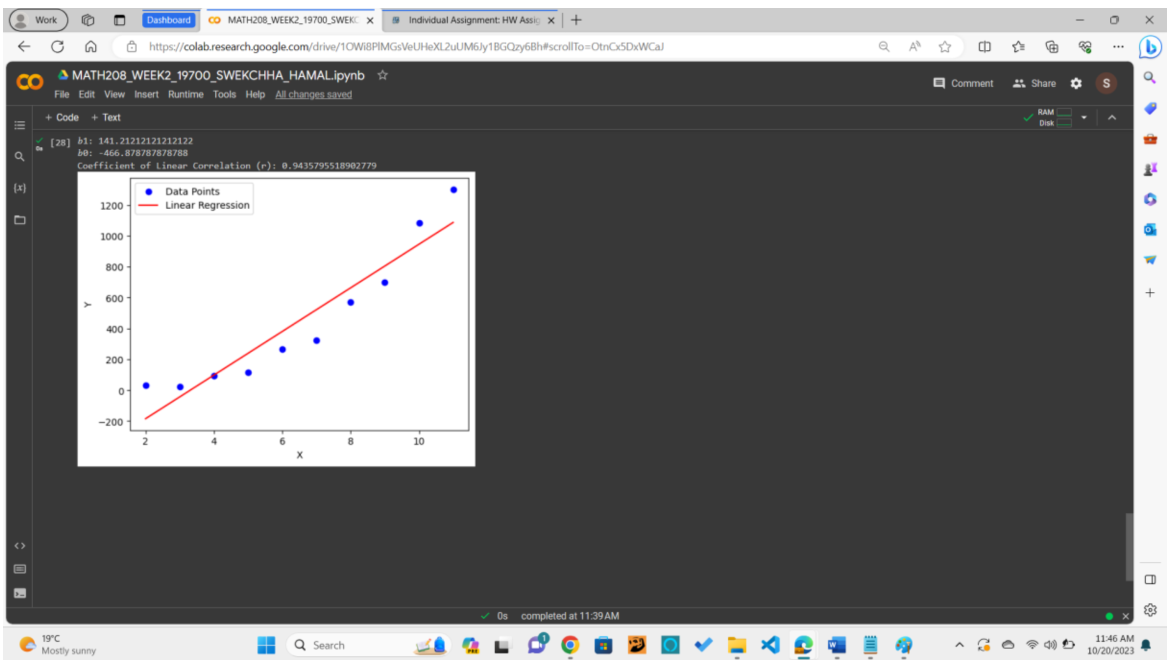


3. Given the following dataset, write the program to fit it by linear regression showing the values of b_1, b_0 and coefficient of linear correlation r. After that, please plot the curve of X vs Y and straight fitting line. Can we draw the conclusion that linear model is good for the dataset if the value 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

Ans:





Swekchha Hamal, 19700.