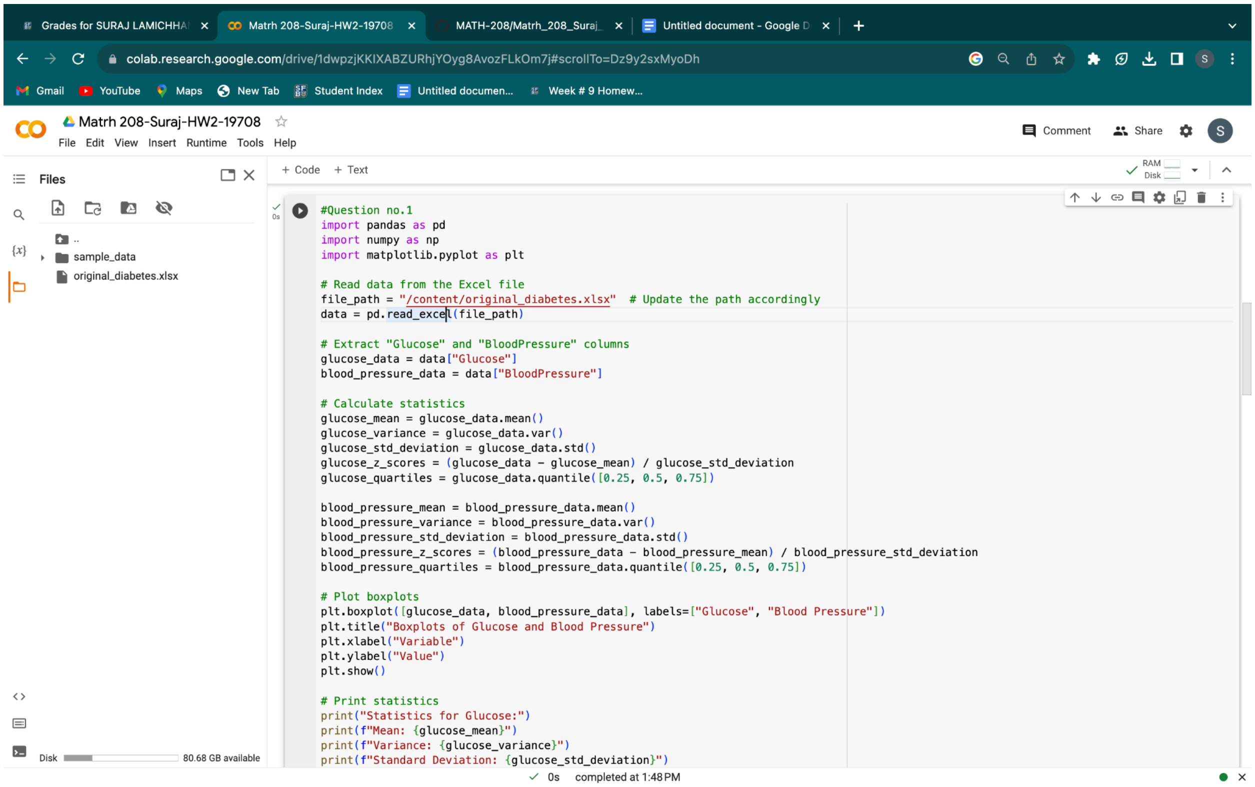
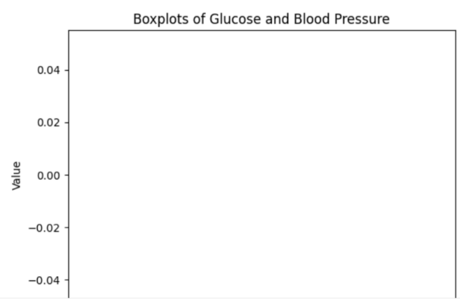
SURAJ LAMICHHANE
19708
MATH 208
Screenshot of Output:

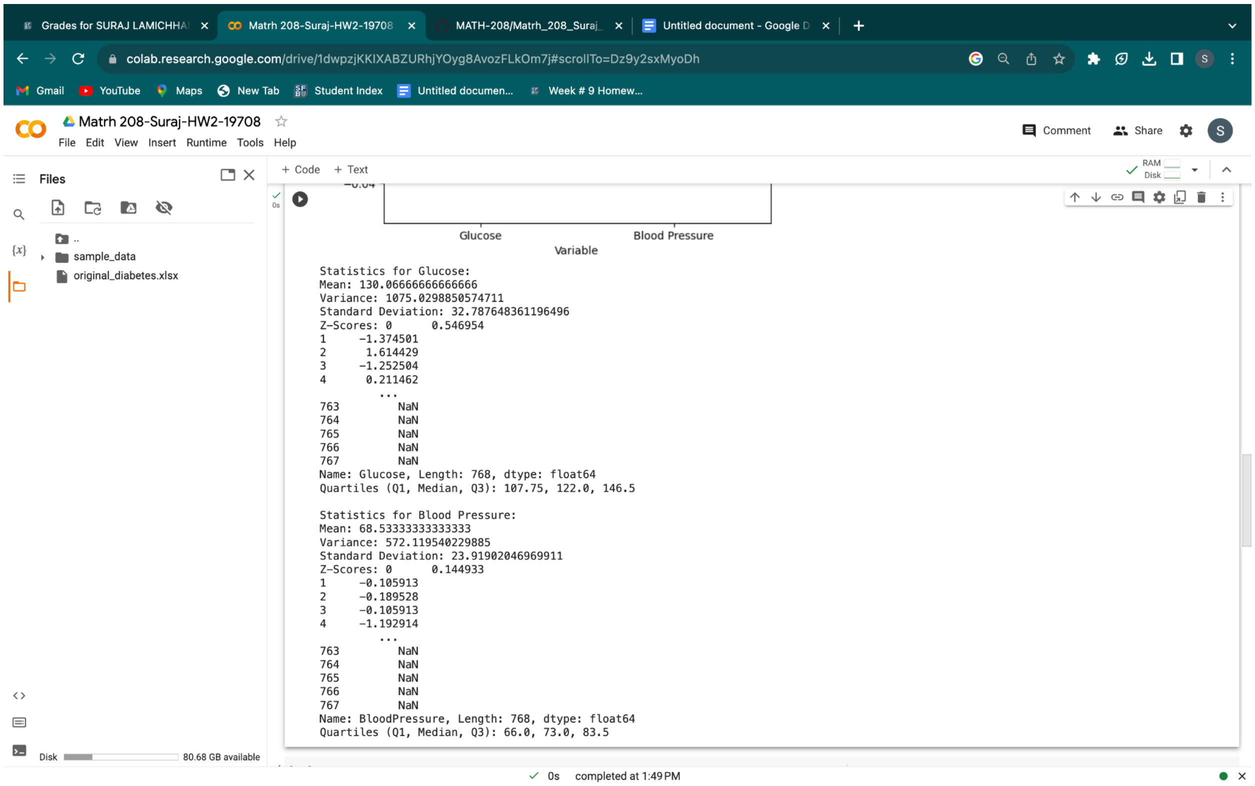
Qno.1)



```python
plt.xlabel("Variable")
plt.ylabel("Value")
plt.show()

# Print statistics
print("Statistics for Glucose:")
print(f"Mean: {glucose_mean}")
print(f"Variance: {glucose_variance}")
print(f"Standard Deviation: {glucose_std_deviation}")
print(f"Z-Scores: {glucose_z_scores}")
print(f"Quartiles (Q1, Median, Q3): {glucose_quartiles[0.25]}, {glucose_quartiles[0.5]}, {glucose_quartiles[0.75]}")

print("\nStatistics for Blood Pressure:")
print(f"Mean: {blood_pressure_mean}")
print(f"Variance: {blood_pressure_variance}")
print(f"Standard Deviation: {blood_pressure_std_deviation}")
print(f"Z-Scores: {blood_pressure_z_scores}")
print(f"Quartiles (Q1, Median, Q3): {blood_pressure_quartiles[0.25]}, {blood_pressure_quartiles[0.5]}, {blood_pressure_quartiles[0.75]}")
```



completed at 1:48 PM


Matrh 208-Suraj-HW2-19708
File Edit View Insert Runtime Tools Help
Files
sample_data
original_diabetes.xlsx
Glucose
Blood Pressure
Variable
Statistics for Glucose:
Mean: 130.06666666666666
Variance: 1075.0298850574711
Standard Deviation: 32.787648361196496
Z-Scores: 0   0.546954
1   -1.374501
2   1.614429
3   -1.252504
4   0.211462
         ...
763   NaN
764   NaN
765   NaN
766   NaN
767   NaN
Name: Glucose, Length: 768, dtype: float64
Quartiles (Q1, Median, Q3): 107.75, 122.0, 146.5
Statistics for Blood Pressure:
Mean: 68.53333333333333
Variance: 572.119540229885
Standard Deviation: 23.91902046969911
Z-Scores: 0   0.144933
1   -0.105913
2   -0.189528
3   -0.105913
4   -1.192914
         ...
763   NaN
764   NaN
765   NaN
766   NaN
767   NaN
Name: BloodPressure, Length: 768, dtype: float64
Quartiles (Q1, Median, Q3): 66.0, 73.0, 83.5
Disk 80.68 GB available
0s completed at 1:49 PM

Qno.2)

Matrh 208-Suraj-HW2-19708

```python
#Question no.2
import random
import statistics

# Function to generate 'n' random numbers within the specified range
def generate_random_numbers(n, min_value, max_value):
    return [random.uniform(min_value, max_value) for _ in range(n)]

# Function to verify Chebyshev's inequality
def verify_Chebyshev_ineq(lst, k):
    mean = statistics.mean(lst)
    std_deviation = statistics.stdev(lst)

    lower_bound = mean - k * std_deviation
    upper_bound = mean + k * std_deviation

    count_within_range = sum(1 for x in lst if lower_bound <= x <= upper_bound)

    return count_within_range

# Testcases
n = 50
min_value = -20
max_value = 20
lst = generate_random_numbers(n, min_value, max_value)

# Test Chebyshev's inequality for different values of k
ks = [1, 2**0.5, 1.5, 2, 3]

for k in ks:
    cnt = verify_Chebyshev_ineq(lst, k)
    prob = 1 - 1 / (k**2)
    print(f"When k = {k}, P(|X-u|<k*sd) = {cnt/n:.2f} ; 1-1/(k^2) = {prob:.16f}")
    if cnt / n >= prob:
        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")
```

0s completed at 1:50PM

Matrh 208-Suraj-HW2-19708

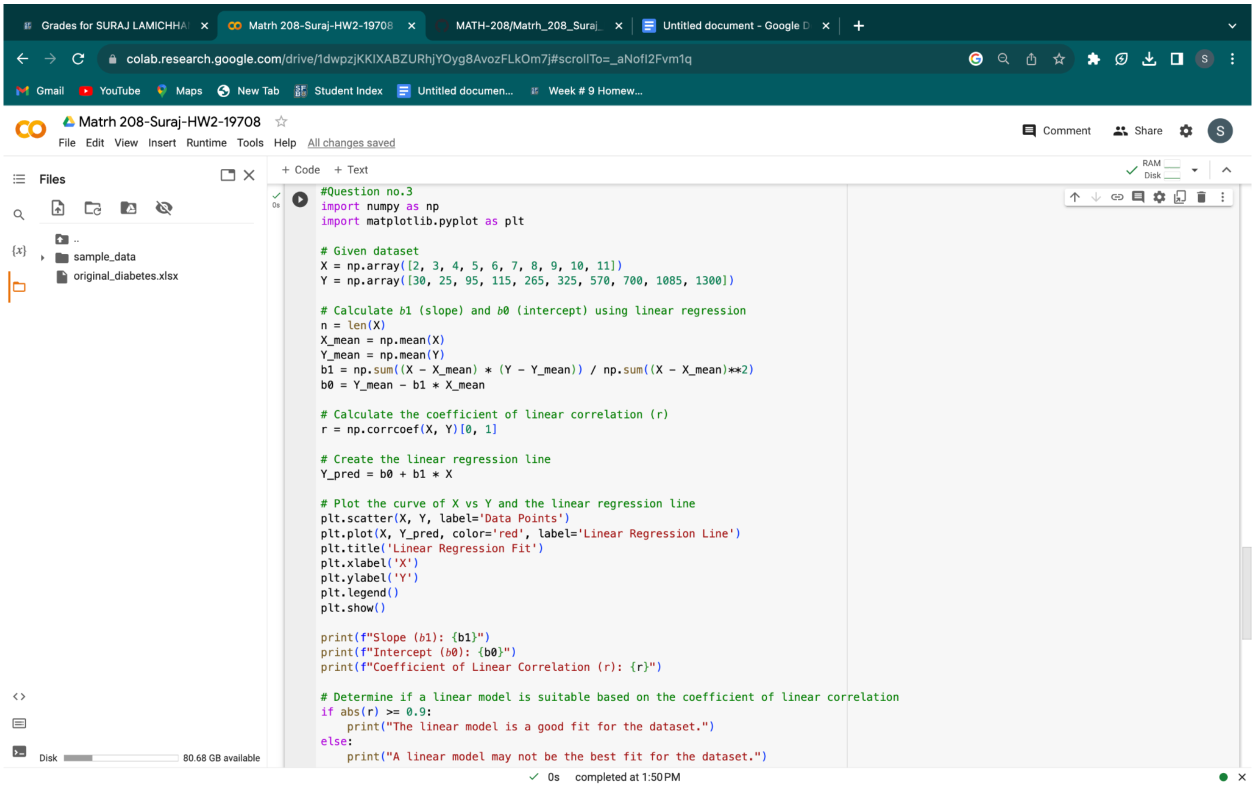
```
When k = 1, P(|X-u|<k*sd) = 0.58 ; 1-1/(k^2) = 0.0000000000000000
When k = 1, P(|X-u|<k*sd) >= 1-1/(k^2) is True
When k = 1.4142135623730951, P(|X-u|<k*sd) = 0.82 ; 1-1/(k^2) = 0.5000000000000001
When k = 1.4142135623730951, P(|X-u|<k*sd) >= 1-1/(k^2) is True
When k = 1.5, P(|X-u|<k*sd) = 0.88 ; 1-1/(k^2) = 0.5555555555555556
When k = 1.5, P(|X-u|<k*sd) >= 1-1/(k^2) is True
When k = 2, P(|X-u|<k*sd) = 1.00 ; 1-1/(k^2) = 0.7500000000000000
When k = 2, P(|X-u|<k*sd) >= 1-1/(k^2) is True
When k = 3, P(|X-u|<k*sd) = 1.00 ; 1-1/(k^2) = 0.8888888888888888
When k = 3, P(|X-u|<k*sd) >= 1-1/(k^2) is True
```

[35]

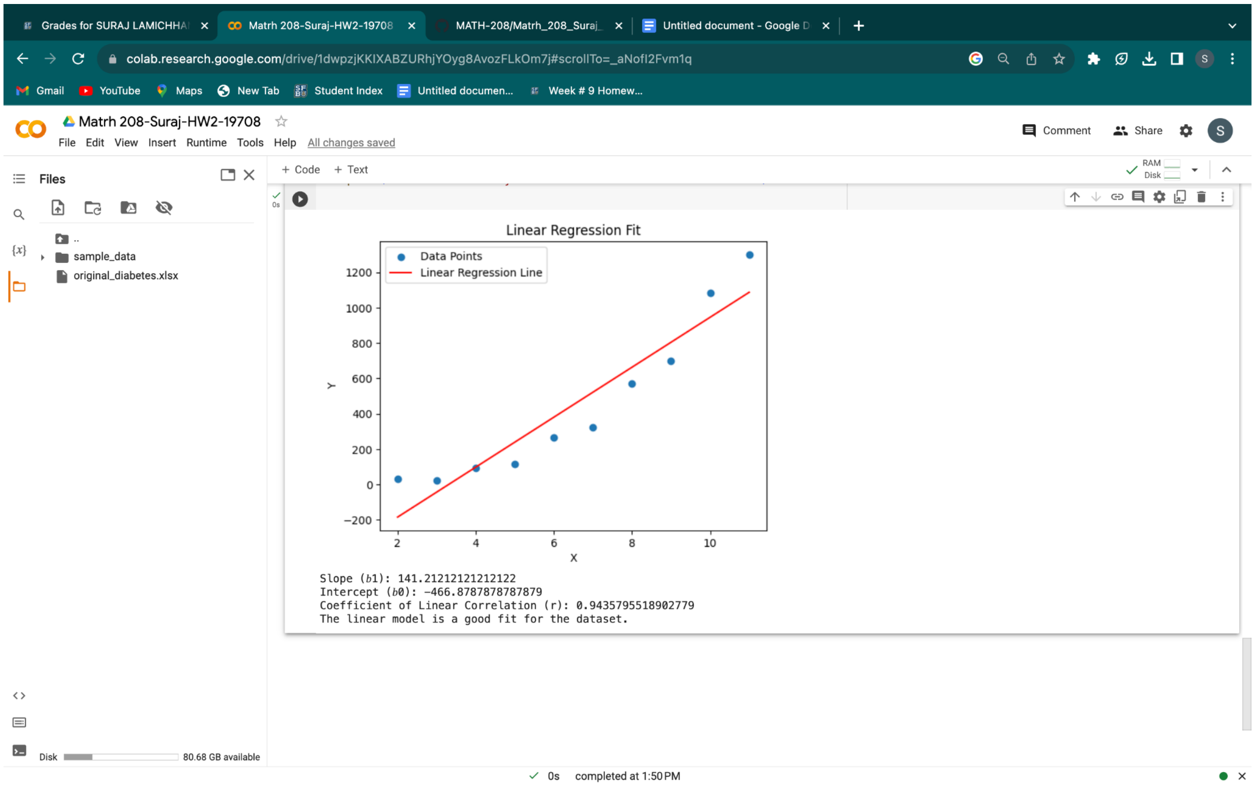
```python
#Question no.1
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Read data from the Excel file
file_path = "/content/original_diabetes.xlsx"  # Update the path accordingly
data = pd.read_excel(file_path)

# Extract "Glucose" and "BloodPressure" columns
glucose_data = data["Glucose"]
blood_pressure_data = data["BloodPressure"]

# Calculate statistics
glucose_mean = glucose_data.mean()
glucose_variance = glucose_data.var()
glucose_std_deviation = glucose_data.std()
glucose_z_scores = (glucose_data - glucose_mean) / glucose_std_deviation
glucose_quartiles = glucose_data.quantile([0.25, 0.5, 0.75])

blood_pressure_mean = blood_pressure_data.mean()
blood_pressure_variance = blood_pressure_data.var()
blood_pressure_std_deviation = blood_pressure_data.std()
blood_pressure_z_scores = (blood_pressure_data - blood_pressure_mean) / blood_pressure_std_deviation
blood_pressure_quartiles = blood_pressure_data.quantile([0.25, 0.5, 0.75])
```

0s completed at 1:50PM

Qno.3)




```python
#Question no.3
import numpy as np
import matplotlib.pyplot as plt

# Given 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])

# Calculate b1 (slope) and b0 (intercept) using linear regression
n = len(X)
X_mean = np.mean(X)
Y_mean = np.mean(Y)
b1 = np.sum((X - X_mean) * (Y - Y_mean)) / np.sum((X - X_mean)**2)
b0 = Y_mean - b1 * X_mean

# Calculate the coefficient of linear correlation (r)
r = np.corrcoef(X, Y)[0, 1]

# Create the linear regression line
Y_pred = b0 + b1 * X

# Plot the curve of X vs Y and the linear regression line
plt.scatter(X, Y, label='Data Points')
plt.plot(X, Y_pred, color='red', label='Linear Regression Line')
plt.title('Linear Regression Fit')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()

print(f"Slope (b1): {b1}")
print(f"Intercept (b0): {b0}")
print(f"Coefficient of Linear Correlation (r): {r}")

# Determine if a linear model is suitable based on the coefficient of linear correlation
if abs(r) >= 0.9:
    print("The linear model is a good fit for the dataset.")
else:
    print("A linear model may not be the best fit for the dataset.")
```



Grades for SURAJ LAMICHHA
Matrh 208-Suraj-HW2-19708
MATH-208/Matrh_208_Suraj_
Untitled document - Google D
colab.research.google.com/drive/1dwpzjKKIXABZURhjYOyg8AvozFLkOm7j#scrollTo=_aNofI2Fvm1q
Gmail
YouTube
Maps
New Tab
Student Index
Untitled documen...
Week # 9 Homew...
Matrh 208-Suraj-HW2-19708
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
sample_data
original_diabetes.xlsx
+ Code + Text
RAM
Disk
Linear Regression Fit
Data Points
Linear Regression Line
Y
X
Slope (b1): 141.21212121212122
Intercept (b0): -466.8787878787879
Coefficient of Linear Correlation (r): 0.9435795518902779
The linear model is a good fit for the dataset.
Disk
80.68 GB available
0s
completed at 1:50 PM