imports

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import glob  # Import glob to list CSV files
import pandas as pd
from tabulate import tabulate
from scipy.stats import norm

from google.colab import drive

drive.mount('/content/drive')
```

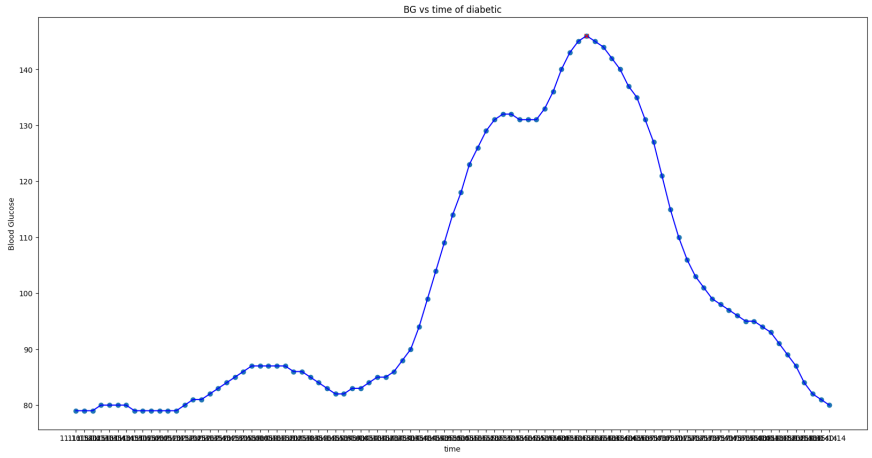
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

import the datas

```
folder_path = '/content/drive/MyDrive/Colab Notebooks/data_for_diabetes'
diabetic = glob.glob(folder_path + 'case *.csv')

for file in diabetic:

    df = pd.read_csv(file)

nondiabetic = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/data for no diabetes.csv')
```

Blood glucose level of person with Diabetic vs time

```
all_data = []
diabetic = glob.glob(folder_path + '/case*.csv')
for file in diabetic:
    df = pd.read_csv(file)
    all_data.append(df)
df = all_data[-1]

plt.figure(figsize=(20, 10))
x = df.iloc[134:225 , 1]
y = df.iloc[134:225 , 2]

plt.scatter(x, y)
plt.title('BG vs time of diabetic')
plt.xlabel('time')
plt.ylabel('Blood Glucose')
plt.plot(x, y, color='blue')

max_y = y.max()
max_x = x[y.idxmax()]
plt.plot(max_x, max_y, "x", color='red')

plt.show()
```



person with diabetic analysis

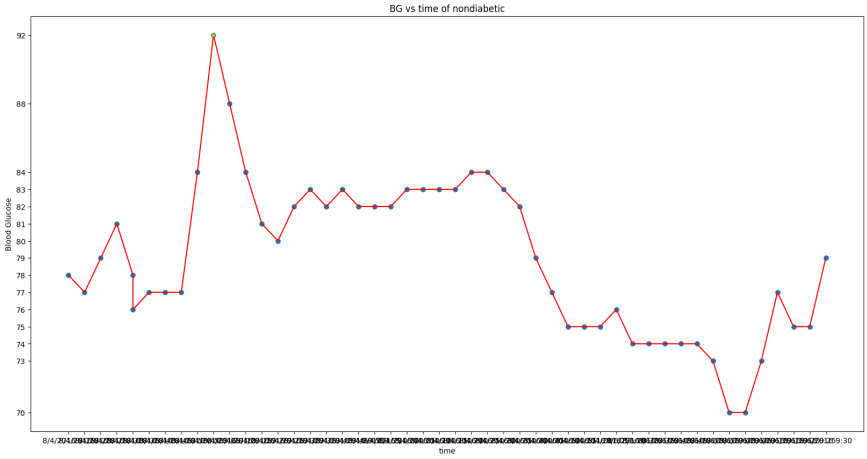
```
df_diabetic = df.iloc[:, 2]
#134:225
mean_diabetic = df_diabetic.mean()
median_diabetic = df_diabetic.median()
range_diabetic = df_diabetic.max() - df_diabetic.min()
std_diabetic = df_diabetic.std()


stats_df = pd.DataFrame({
    'Statistic Diabetic': ['Mean', 'Median', 'Range', 'Standard Deviation'],
    'Value': [mean_diabetic, median_diabetic, range_diabetic, std_diabetic]
})

print(tabulate(stats_df, headers='keys', tablefmt='fancy_grid'))
```

	Statistic Diabetic	Value
0	Mean	98.6332
1	Median	94
2	Range	86
3	Standard Deviation	19.712

non-diabetic vs time graph

```
plt.figure(figsize=(20, 10))
x = nondiabetic.iloc[1:50, 2]
y = data = pd.to_numeric(nondiabetic.iloc[1:50, 3], errors='coerce')
plt.scatter(x, y)

plt.title('BG vs time of nondiabetic')
plt.xlabel('time')
plt.ylabel('Blood Glucose')



sorted_y = np.sort(y.unique())
plt.yticks(sorted_y)

plt.plot(x, y, color='red')

max_y = y.max()
max_x = x[y.idxmax()]


plt.plot(max_x, max_y, "x", color='yellow')
```

```
[<matplotlib.lines.Line2D at 0x7cd687f60880>]
```



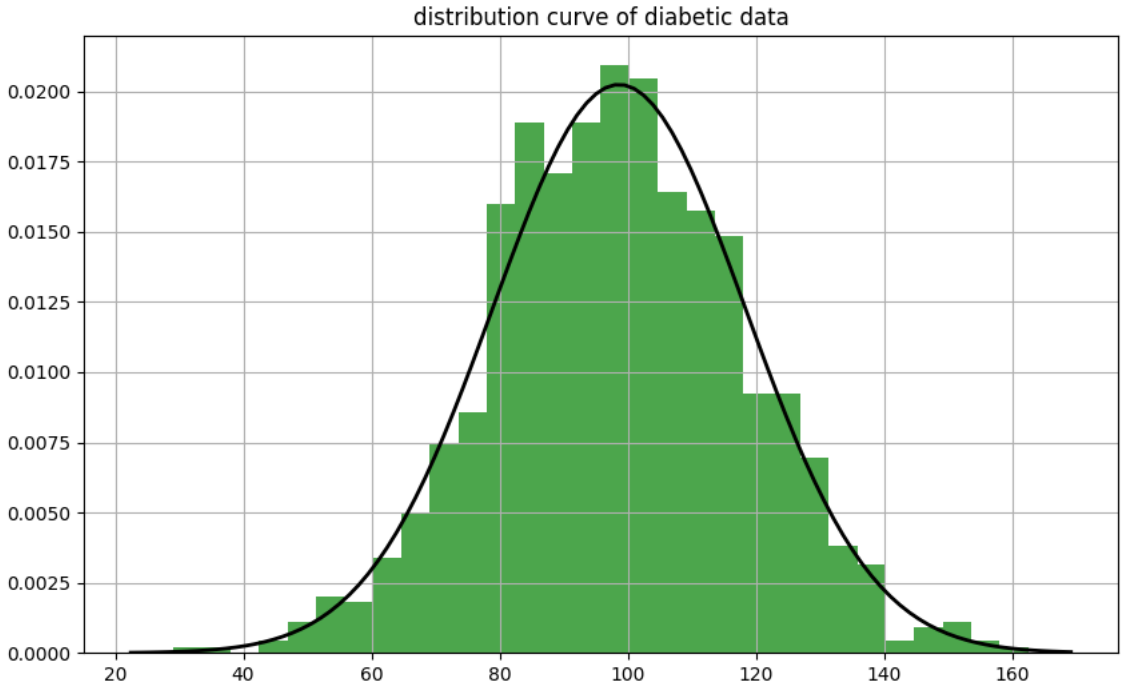
non-diabetic analysis

```
df_nondiabetic = pd.to_numeric(nondiabetic.iloc[:, 3], errors='coerce') #from stackoverflow, this ignores the non-numeric number
#1:50
mean_nondiabetic = df_nondiabetic.mean()
mean_nondiabetic = df_nondiabetic.mean()
median_nondiabetic = df_nondiabetic.median()
range_nondiabetic = df_nondiabetic.max() - df_nondiabetic.min()
std_nondiabetic = df_nondiabetic.std()


stats_df = pd.DataFrame({
    'Statistic nondiabetic': ['Mean', 'Median', 'Range', 'Standard Deviation'],
    'Value': [mean_nondiabetic, median_nondiabetic, range_nondiabetic, std_nondiabetic]
})
print(tabulate(stats_df, headers='keys', tablefmt='fancy_grid'))
```

	Statistic nondiabetic	Value
0	Mean	82.9286
1	Median	82
2	Range	43
3	Standard Deviation	8.53051

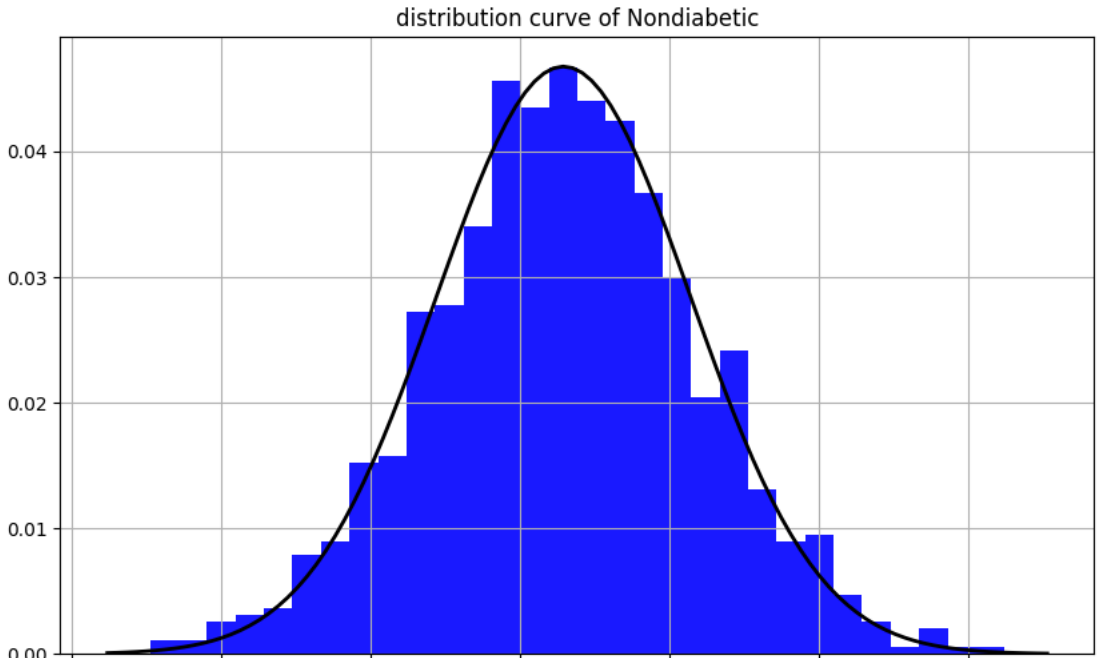
distribution graph of diabetic data

```
mean = mean_diabetic
std_dev = std_diabetic


data = np.random.normal(mean, std_dev, 1000)


plt.figure(figsize=(10, 6))
plt.hist(data, bins=30, density=True, alpha=0.7, color='g')
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
pdf_values = norm.pdf(x, mean, std_dev)

plt.plot(x, pdf_values, 'k', linewidth=2) #the normal dist. curve


plt.title("distribution curve of diabetic data" )
plt.grid(True)
plt.show()
```



distribution for nondiabetic

```
mean = mean_nondiabetic
std_dev = std_nondiabetic


data = np.random.normal(mean, std_dev, 1000)


plt.figure(figsize=(10, 6))

plt.hist(data, bins=30, density=True, alpha=0.9, color='b')
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)

pdf_values = norm.pdf(x, mean, std_dev)

plt.plot(x, pdf_values, 'k', linewidth=2) #the normal dist. curve

plt.title("distribution curve of Nondiabetic")

plt.grid(True)
plt.show()
```



finding confidence interval

```
import scipy.stats as stats

x1_bar = mean_diabetic
x2_bar = mean_nondiabetic
n1 = 208
n2 = 57

# Calculate the standard error (SE)
SE = ((std_diabetic**2 / n1) + (std_nondiabetic**2 / n2)) ** 0.5
SE
# Degrees of freedom using Welch's formula
df = n2 - 1

# Critical value from the t-distribution for a 95% confidence interval
t_critical = stats.t.ppf(0.95, df)  # Two-tailed test

# Margin of error (ME)
ME = t_critical * SE

# Confidence interval (CI)
CI_lower = (x1_bar - x2_bar) - ME
CI_upper = (x1_bar - x2_bar) + ME


confidence_interval = pd.DataFrame({
    'confidence interval' : ['lower bound', 'upper bound'],
    'Value': [CI_lower, CI_upper]
})
print(tabulate(confidence_interval, headers='keys', tablefmt='fancy_grid'))
```

	confidence interval	Value
0	lower bound	12.7386
1	upper bound	18.6705

calculating P value

```
import scipy.stats as stats

T_value = (x1_bar - x2_bar)/SE
T_value

p_value = 2 * (1 - stats.t.cdf(abs(T_value), df))


p = pd.DataFrame({
    'p value of two tailed test' : ['P value', 'conclusion:'],
    'Value': [p_value, 'p value < 0.05 thus rejecting the null hypothesis, the datas are significantly different']
})
```

```
print(tabulate(p, headers='keys', tablefmt='fancy_grid'))
```

	p value of two tailed test	Value
0	P value	3.08575387464316e-12
1	conclusion:	p value < 0.05 thus rejecting the null hypothesis, the datas are significantly different