In [1]:
```
# SOLUTION:

# Solution for correcting data quality issues
# Debug data first!

# In this dataset, we have lot of observations that have missing values
# Missing values are represented using 0s
# We need to impute values; one option is to find out mean for every class and use that as a substitute
# for missing values
# With these changes,the model F1 score improves from 0.65 to 0.81

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Diabetes Binary Classification Dataset

Input Features: 'preg_count', 'glucose_concentration', 'diastolic_bp', 'triceps_skin_fold_thickness', two_hr_serum_insulin', 'bmi', 'diabetes_pedi', 'age'

Target Feature: 'diabetes_class'

Objective: Predict diabetes_class for given input features

Data Source: https://archive.ics.uci.edu/ml/datasets/pima+indians+diabetes

In [2]:
```
columns = ['diabetes_class', 'preg_count', 'glucose_concentration', 'diastolic_bp',
       'triceps_skin_fold_thickness', 'two_hr_serum_insulin', 'bmi',
       'diabetes_pedi', 'age']
```

In [3]:
```
df = pd.read_csv('pima_indians_diabetes_all.csv')
```

In [4]:
```
# Look for any columns that have NA
df.isna().any(axis=0)
```

Out[4]:
```
preg_count                    False
glucose_concentration         False
diastolic_bp                  False
triceps_skin_fold_thickness   False
two_hr_serum_insulin          False
bmi                           False
diabetes_pedi                 False
age                           False
diabetes_class                False
dtype: bool
```

In [5]:
```
df.describe()
```

Out[5]:

	preg_count	glucose_concentration	diastolic_bp	triceps_skin_fold_thickness	two_hr_serum_insulin	bmi	diabetes_pedi	
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000

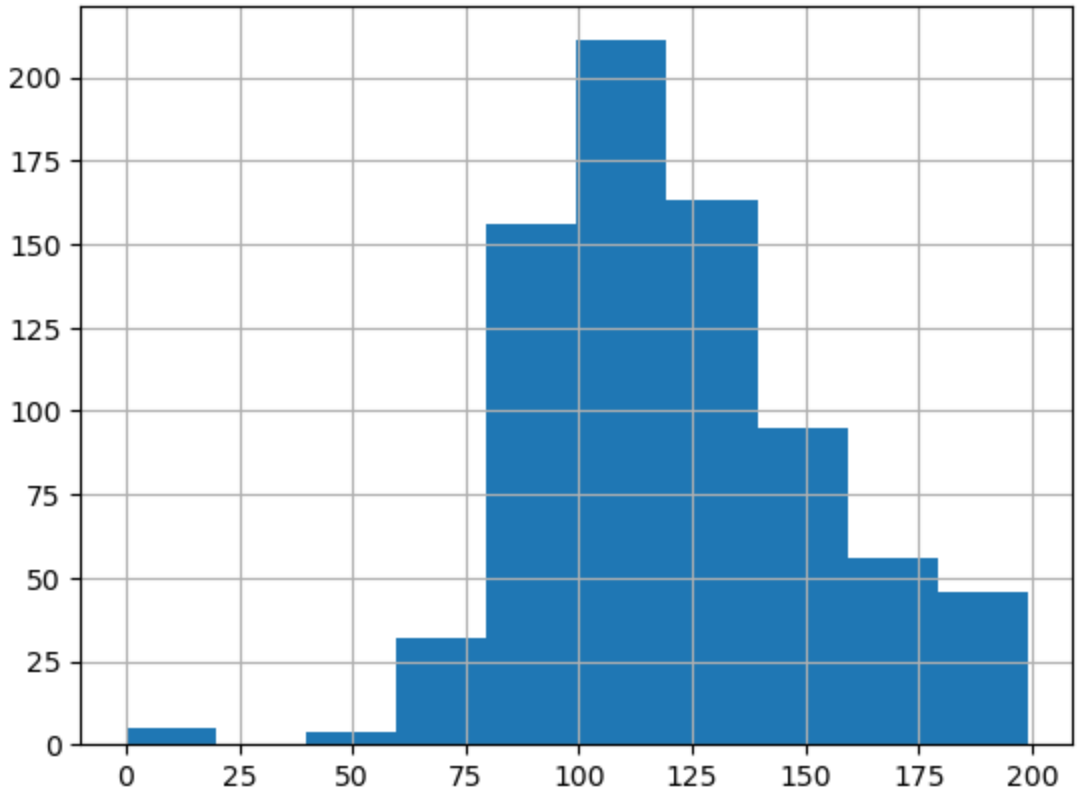
In [7]:
```
#DWB#
print(df.describe())
```

```
       preg_count  glucose_concentration  diastolic_bp
count  768.000000             768.000000    768.000000  \
mean     3.845052             120.894531     69.105469
std      3.369578              31.972618     19.355807
min      0.000000               0.000000      0.000000
25%      1.000000              99.000000     62.000000
50%      3.000000             117.000000     72.000000
75%      6.000000             140.250000     80.000000
max     17.000000             199.000000    122.000000

       triceps_skin_fold_thickness  two_hr_serum_insulin         bmi
count                   768.000000            768.000000  768.000000  \
mean                     20.536458             79.799479   31.992578
std                      15.952218            115.244002    7.884160
min                       0.000000              0.000000    0.000000
25%                       0.000000              0.000000   27.300000
50%                      23.000000             30.500000   32.000000
75%                      32.000000            127.250000   36.600000
max                      99.000000            846.000000   67.100000

       diabetes_pedi         age  diabetes_class
count     768.000000  768.000000      768.000000
mean        0.471876   33.240885        0.348958
std         0.331329   11.760232        0.476951
min         0.078000   21.000000        0.000000
25%         0.243750   24.000000        0.000000
50%         0.372500   29.000000        0.000000
75%         0.626250   41.000000        1.000000
max         2.420000   81.000000        1.000000
```

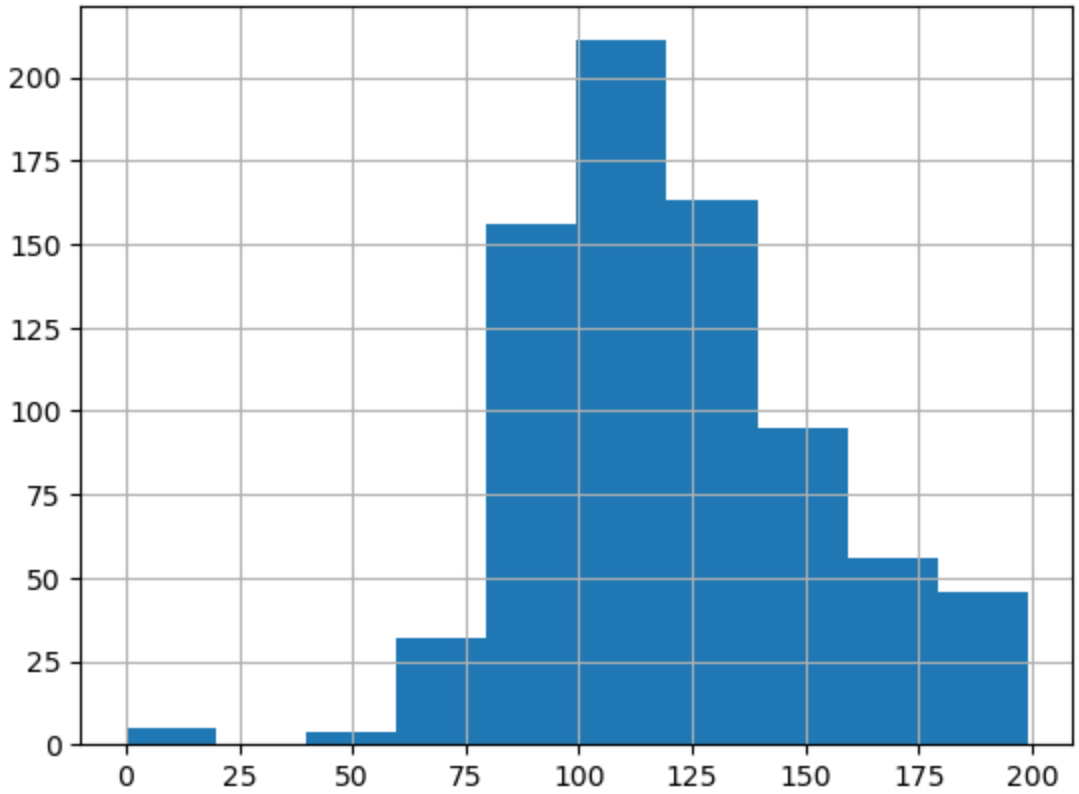
In [8]:
```
#DWB# Investigating data.
df_0_glucose_cncntrtn = df[df['glucose_concentration'] == 0]
print(df_0_glucose_cncntrtn.head())
```

```
     preg_count  glucose_concentration  diastolic_bp
75            1                      0            48  \
182           1                      0            74
342           1                      0            68
349           5                      0            80
502           6                      0            68

     triceps_skin_fold_thickness  two_hr_serum_insulin   bmi  diabetes_pedi
75                            20                     0  24.7          0.140  \
182                           20                    23  27.7          0.299
342                           35                     0  32.0          0.389
349                           32                     0  41.0          0.346
502                           41                     0  39.0          0.727

     age  diabetes_class
75    22               0
182   21               0
342   22               0
349   37               1
502   41               1
```

In [9]:
```
df['glucose_concentration'].hist()
plt.show()
```



In [10]:
```
#DWB#  Attempt at mean with both broups combined,
#DWB#+ need deep copy now.
df_no_group = df.copy(deep=True)
df_no_group['glucose_concentration'].hist()
plt.show()
```



In [11]:
```
# Find Summary Statistics for Each Class
# Impute values based on class
# https://stackoverflow.com/questions/19966018/pandas-filling-missing-values-by-mean-in-each-group
group_class = df.groupby('diabetes_class')
```

In [12]:
```
# First few rows of each group
group_class.head(2)
```

Out[12]:

	preg_count	glucose_concentration	diastolic_bp	triceps_skin_fold_thickness	two_hr_serum_insulin	bmi	diabetes_pedi	age	diabetes_cla
0	6	148	72	35	0	33.6	0.627	50	
1	1	85	66	29	0	26.6	0.351	31	
2	8	183	64	0	0	23.3	0.672	32	
3	1	89	66	23	94	28.1	0.167	21	

In [13]:
```
#DWB#
print(group_class.head(2))
```

```
   preg_count  glucose_concentration  diastolic_bp
0           6                    148            72  \
1           1                     85            66
2           8                    183            64
3           1                     89            66

   triceps_skin_fold_thickness  two_hr_serum_insulin   bmi  diabetes_pedi
0                           35                     0  33.6          0.627  \
1                           29                     0  26.6          0.351
2                            0                     0  23.3          0.672
3                           23                    94  28.1          0.167

   age  diabetes_class
0   50               1
1   31               0
2   32               1
3   21               0
```

In [14]:
```
# Attribute Mean value is different for each group
group_class.mean()
```

Out[14]:

diabetes_class	preg_count	glucose_concentration	diastolic_bp	triceps_skin_fold_thickness	two_hr_serum_insulin	bmi	diabetes_pedi
0	3.298000	109.980000	68.184000	19.664000	68.792000	30.304200	0.429734
1	4.865672	141.257463	70.824627	22.164179	100.335821	35.142537	0.550500

In [17]:
```
#DWB#
df.mean()
```

Out[17]:
```
preg_count                    3.845052
glucose_concentration       120.894531
diastolic_bp                 69.105469
triceps_skin_fold_thickness  20.536458
two_hr_serum_insulin         79.799479
bmi                          31.992578
diabetes_pedi                 0.471876
age                          33.240885
diabetes_class                0.348958
dtype: float64
```

In [15]:
```
#DWB#
print(group_class.mean())
```

```
                preg_count  glucose_concentration  diastolic_bp
diabetes_class
0                 3.298000             109.980000     68.184000  \
1                 4.865672             141.257463     70.824627

                triceps_skin_fold_thickness  two_hr_serum_insulin        bmi
diabetes_class
0                                 19.664000             68.792000  30.304200  \
1                                 22.164179            100.335821  35.142537

                diabetes_pedi        age
diabetes_class
0                    0.429734  31.190000
1                    0.550500  37.067164
```

In [18]:
```
df['diabetes_class'].value_counts()
```

Out[18]:
```
diabetes_class
0    500
1    268
Name: count, dtype: int64
```

In [19]:
```
# For each group, use group level averages to fill missing values
df['glucose_concentration'] = group_class['glucose_concentration'].transform(lambda x: x.replace(0,x.mean()))
df['diastolic_bp'] = group_class['diastolic_bp'].transform(lambda x: x.replace(0,x.mean()))
df['triceps_skin_fold_thickness'] = group_class['triceps_skin_fold_thickness'].transform(lambda x: x.replace(0,x.mean
df['two_hr_serum_insulin'] = group_class['two_hr_serum_insulin'].transform(lambda x: x.replace(0,x.mean()))
df['bmi'] = group_class['bmi'].transform(lambda x: x.replace(0,x.mean()))
df['diabetes_pedi'] = group_class['diabetes_pedi'].transform(lambda x: x.replace(0,x.mean()))
df['age'] = group_class['age'].transform(lambda x: x.replace(0,x.mean()))
```

In [20]:
```
#DWB# Trying to get rid of the "tell" - doing mean without grouping
df_no_group['glucose_concentration'] = df_no_group['glucose_concentration'].transform(lambda x: x.replace(0,x.mean()
df_no_group['diastolic_bp'] = df_no_group['diastolic_bp'].transform(lambda x: x.replace(0,x.mean()))
df_no_group['triceps_skin_fold_thickness'] = df_no_group['triceps_skin_fold_thickness'].transform(lambda x: x.replace
df_no_group['two_hr_serum_insulin'] = df_no_group['two_hr_serum_insulin'].transform(lambda x: x.replace(0,x.mean()))
df_no_group['bmi'] = df_no_group['bmi'].transform(lambda x: x.replace(0,x.mean()))
df_no_group['diabetes_pedi'] = df_no_group['diabetes_pedi'].transform(lambda x: x.replace(0,x.mean()))
df_no_group['age'] = df_no_group['age'].transform(lambda x: x.replace(0,x.mean()))
```

In [21]:
```
df.head()
```

Out[21]:

	preg_count	glucose_concentration	diastolic_bp	triceps_skin_fold_thickness	two_hr_serum_insulin	bmi	diabetes_pedi	age	diabetes_cla
0	6	148.0	72.0	35.000000	100.335821	33.6	0.627	50	
1	1	85.0	66.0	29.000000	68.792000	26.6	0.351	31	
2	8	183.0	64.0	22.164179	100.335821	23.3	0.672	32	
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21	
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33	

In [22]:
```
#DWB#
print(df.head())
```

```
   preg_count  glucose_concentration  diastolic_bp
0           6                  148.0          72.0  \
1           1                   85.0          66.0
2           8                  183.0          64.0
3           1                   89.0          66.0
4           0                  137.0          40.0

   triceps_skin_fold_thickness  two_hr_serum_insulin   bmi  diabetes_pedi
0                    35.000000            100.335821  33.6          0.627  \
1                    29.000000             68.792000  26.6          0.351
2                    22.164179            100.335821  23.3          0.672
3                    23.000000             94.000000  28.1          0.167
4                    35.000000            168.000000  43.1          2.288

   age  diabetes_class
0   50               1
1   31               0
2   32               1
3   21               0
4   33               1
```

In [23]:
```
#DWB#
df_no_group.head()
```

Out[23]:

	preg_count	glucose_concentration	diastolic_bp	triceps_skin_fold_thickness	two_hr_serum_insulin	bmi	diabetes_pedi	age	diabetes_cla
0	6	148.0	72.0	35.000000	79.799479	33.6	0.627	50	
1	1	85.0	66.0	29.000000	79.799479	26.6	0.351	31	
2	8	183.0	64.0	20.536458	79.799479	23.3	0.672	32	
3	1	89.0	66.0	23.000000	94.000000	28.1	0.167	21	
4	0	137.0	40.0	35.000000	168.000000	43.1	2.288	33	

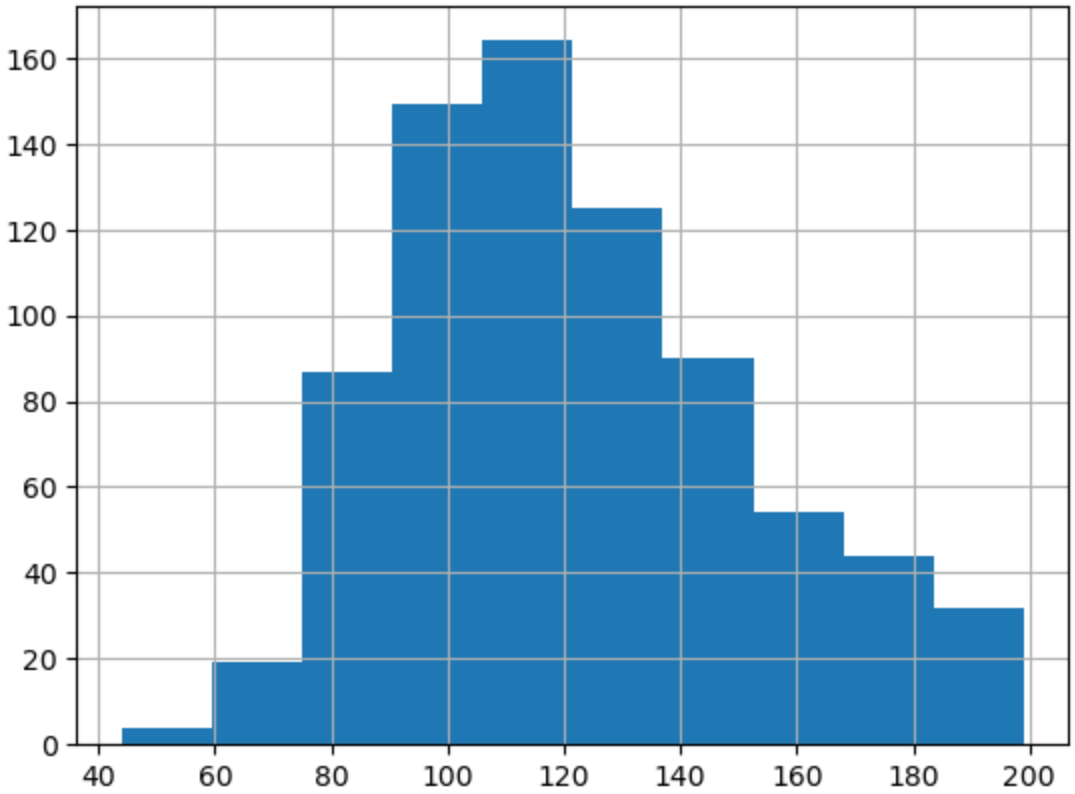
In [24]:
```
#DWB#
print(df_no_group.head())
```

```
   preg_count  glucose_concentration  diastolic_bp
0           6                  148.0          72.0  \
1           1                   85.0          66.0
2           8                  183.0          64.0
3           1                   89.0          66.0
4           0                  137.0          40.0

   triceps_skin_fold_thickness  two_hr_serum_insulin   bmi  diabetes_pedi
0                    35.000000             79.799479  33.6          0.627  \
1                    29.000000             79.799479  26.6          0.351
2                    20.536458             79.799479  23.3          0.672
3                    23.000000             94.000000  28.1          0.167
4                    35.000000            168.000000  43.1          2.288

   age  diabetes_class
0   50               1
1   31               0
2   32               1
3   21               0
4   33               1
```

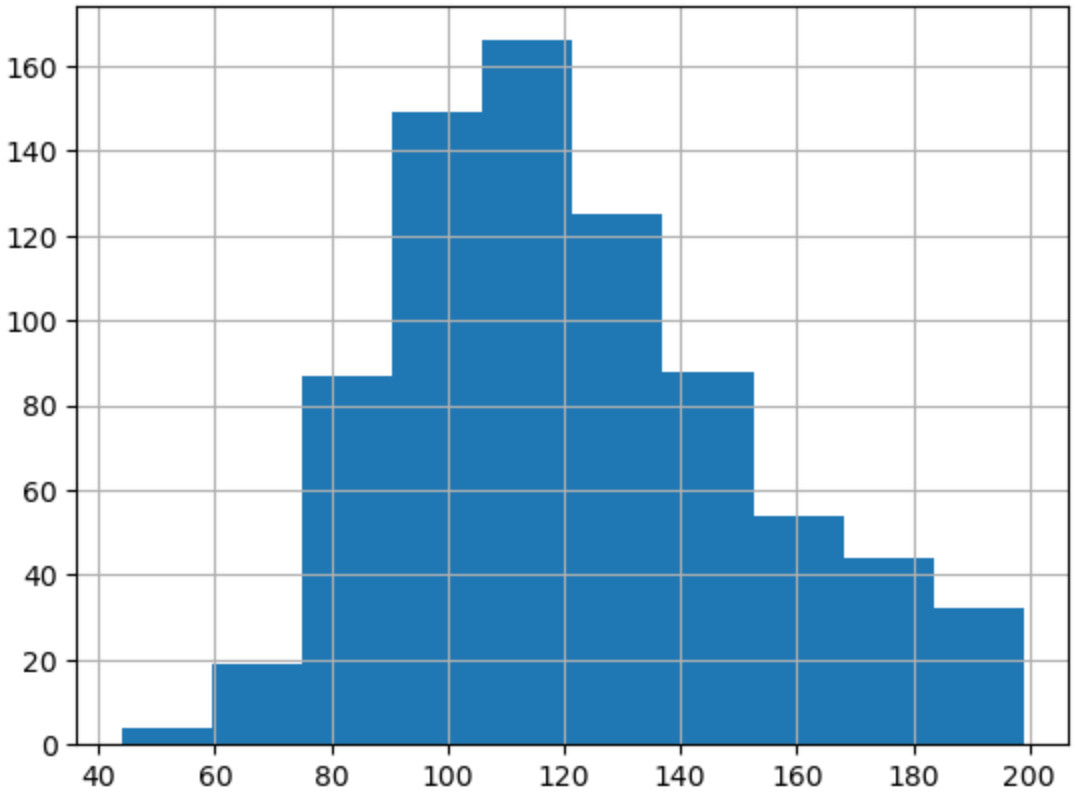
In [30]:
```
#DWB#  Let's see if this made a difference. I'll do histograms,
#DWB#+ below, but I noticed that we do have some differences, e.g.

df_val = df.iloc[2].loc['triceps_skin_fold_thickness']
df_no_group_val = df_no_group.iloc[2].loc['triceps_skin_fold_thickness']

print(f"In df,          the value is: {df_val},")
print(f"In df_no_group, the value is: {df_no_group_val}.")
print("Check: The statement, " +
      f"'They are the same,' is {df_val == df_no_group_val}.")

print()
print()

df_vals = df.loc[[0, 1, 2], 'two_hr_serum_insulin']
df_no_group_vals = df_no_group.loc[[0, 1, 2],'two_hr_serum_insulin']

print(f"df_vals:\n{df_vals}")
print()
print(f"df_no_group_vals:\n{df_no_group_vals}")
```

```
In df,          the value is: 22.16417910447761,
In df_no_group, the value is: 20.536458333333332.
Check: The statement, 'They are the same,' is False.


df_vals:
0    100.335821
1     68.792000
2    100.335821
Name: two_hr_serum_insulin, dtype: float64

df_no_group_vals:
0    79.799479
1    79.799479
2    79.799479
Name: two_hr_serum_insulin, dtype: float64
```

In [31]:
```
#DWB
df['glucose_concentration'].hist()
plt.show()
```



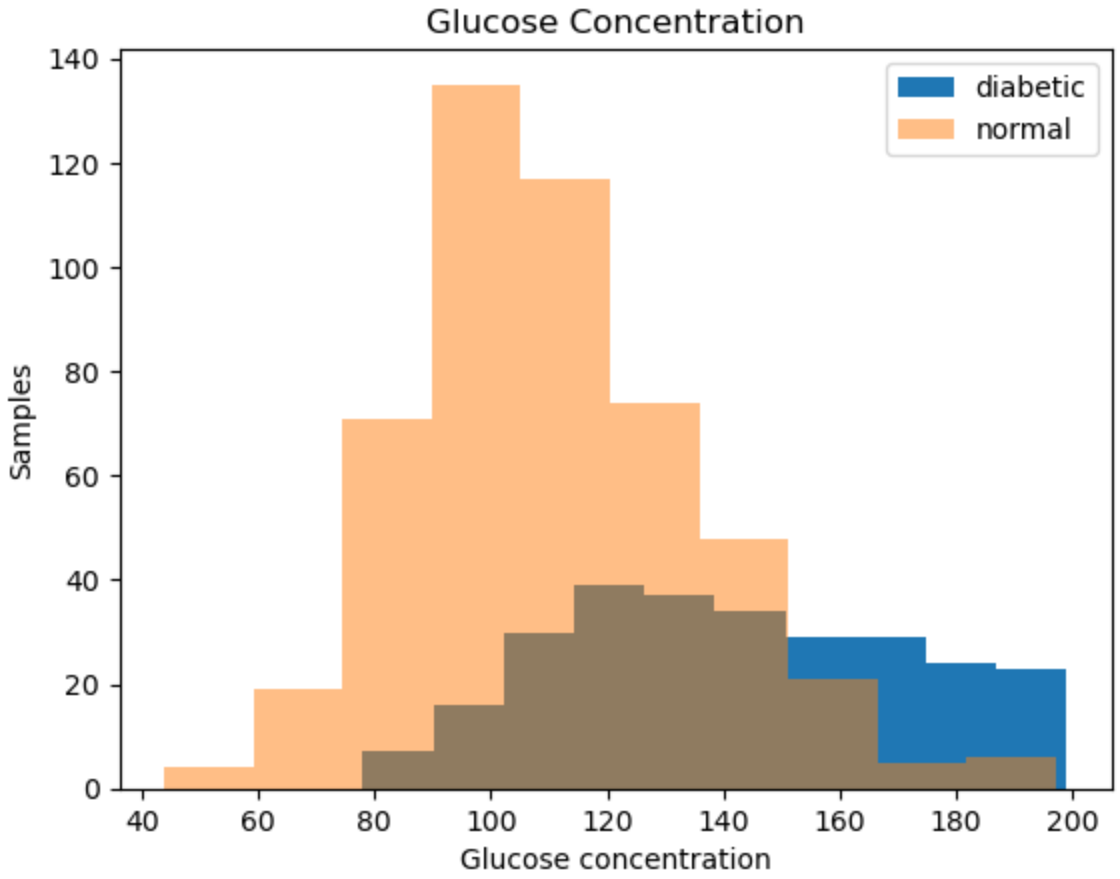
In [32]:
```
#DWB#
df_no_group['glucose_concentration'].hist()
plt.show()
```



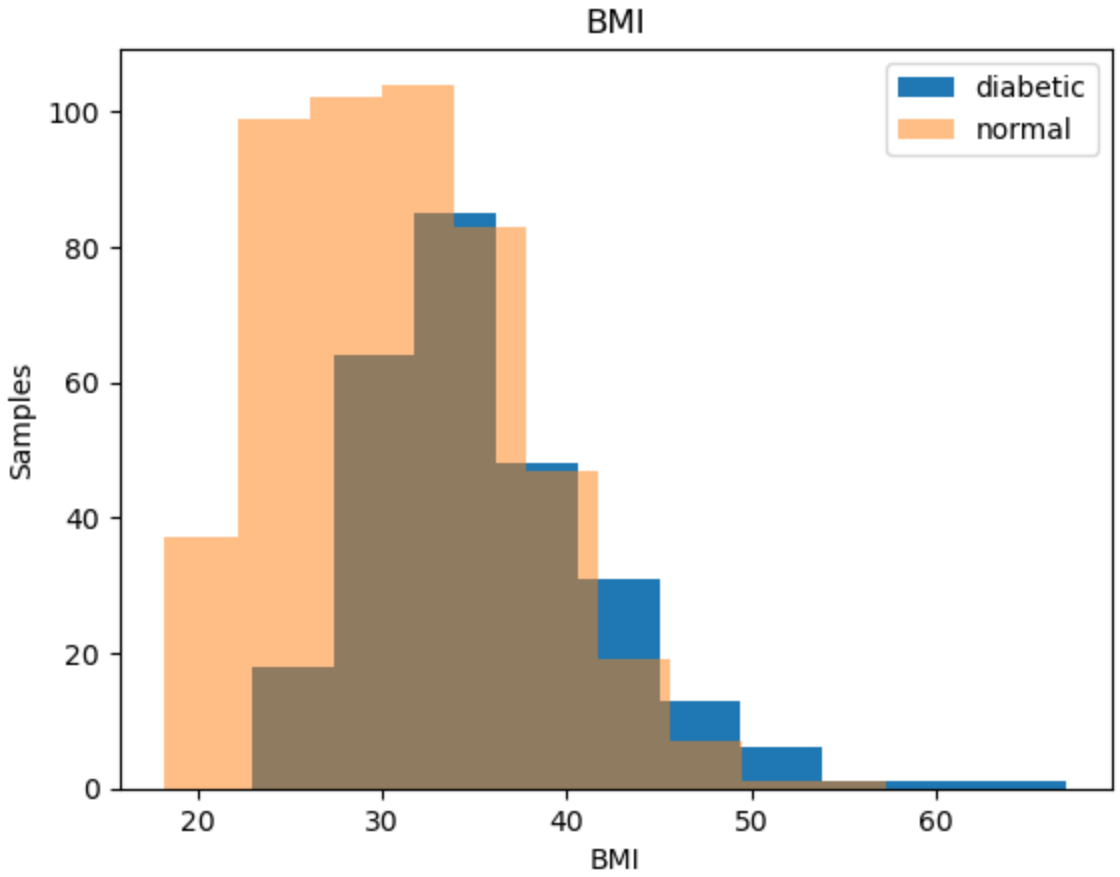
#DWB# It's hard to tell if there's a difference from the histogram. Luckily, we looked at the values, above

In [33]:
```
# Separate diabetic and normal samples
diabetic = df.diabetes_class == 1
normal = df.diabetes_class == 0
```

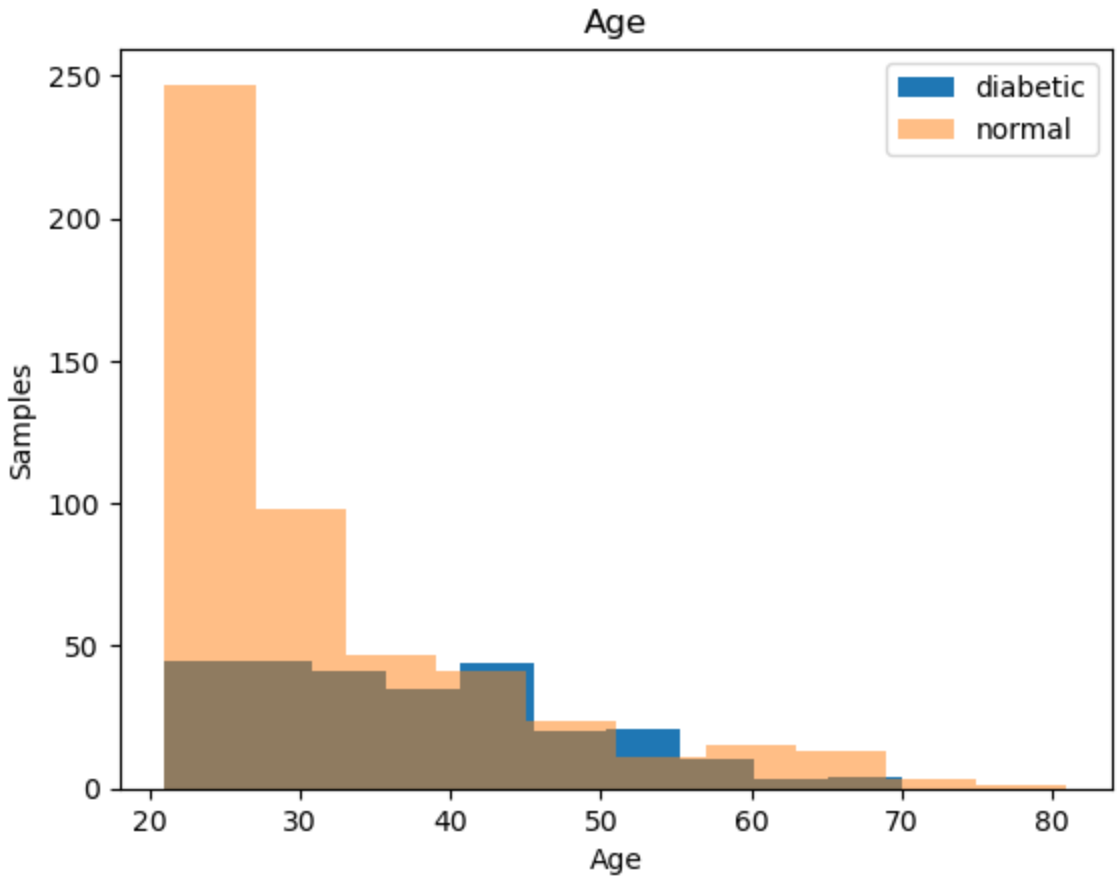
In [34]:
```
# Glucose concentration histogram
plt.hist(df[diabetic].glucose_concentration,label='diabetic')
plt.hist(df[normal].glucose_concentration,alpha=0.5,label='normal')
plt.title('Glucose Concentration')
plt.xlabel('Glucose concentration')
plt.ylabel('Samples')
plt.legend()
plt.show()
```



In [35]:
```
# BMI histogram
plt.hist(df[diabetic].bmi,label='diabetic')
plt.hist(df[normal].bmi,alpha=0.5,label='normal')
plt.title('BMI')
plt.xlabel('BMI')
plt.ylabel('Samples')
plt.legend()
plt.show()
```



In [36]:
```
# Age
plt.hist(df[diabetic].age,label='diabetic')
plt.hist(df[normal].age,alpha=0.5,label='normal')
plt.title('Age')
plt.xlabel('Age')
plt.ylabel('Samples')
plt.legend()
plt.show()
```



Training and Validation Set

Target Variable as first column followed by input features:

'diabetes_class', 'preg_count', 'glucose_concentration', 'diastolic_bp', 'triceps_skin_fold_thickness', 'two_hr_serum_insulin', 'bmi', 'diabetes_pedi', 'age'

Training, Validation files do not have a column header

In [37]:
```
# Training = 70% of the data
# Validation = 30% of the data
```

```
# Randomize the datset
np.random.seed(5)
l = list(df.index)
np.random.shuffle(l)
df = df.iloc[l]
```

In [38]:
```
rows = df.shape[0]
train = int(.7 * rows)
test = rows - train
```

In [39]:
```
rows, train, test
```

Out[39]:
```
(768, 537, 231)
```

In [40]:
```
# Write Training Set
df[:train].to_csv('diabetes_train.csv'
                          ,index=False,index_label='Row',header=False
                          ,columns=columns)
```

In [41]:
```
# Write Validation Set
df[train:].to_csv('diabetes_validation.csv'
                          ,index=False,index_label='Row',header=False
                          ,columns=columns)
```

In [42]:
```
# Write Column List
with open('diabetes_train_column_list.txt','w') as f:
    f.write(','.join(columns))
```

In [ ]:
```

```