In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import style
%matplotlib inline
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('health care diabetes.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
1	1	85	66	29	0	26.6	0.351	
2	8	183	64	0	0	23.3	0.672	
3	1	89	66	23	94	28.1	0.167	
4	0	137	40	35	168	43.1	2.288	

In [4]:
```
df.isna().sum(axis=0)
```

Out[4]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
 3   SkinThickness             768 non-null    int64  
 4   Insulin                   768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

In [6]:
```
new_df = df[df['Outcome']==1]
new_df.head(5)
```

Out[6]:

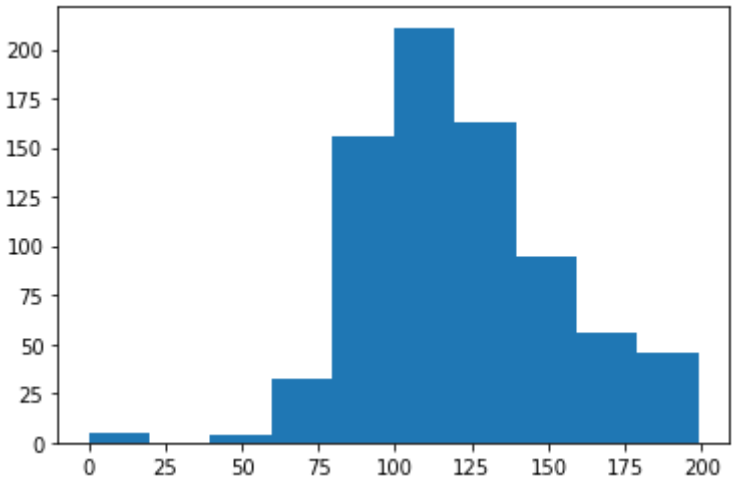
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	A
0	6	148	72	35	0	33.6	0.627	
2	8	183	64	0	0	23.3	0.672	
4	0	137	40	35	168	43.1	2.288	
6	3	78	50	32	88	31.0	0.248	
8	2	197	70	45	543	30.5	0.158	

In [7]:
```
df['Glucose'].value_counts()
```

Out[7]:
```
99     17
100    17
111    14
129    14
125    14
       ..
191     1
177     1
44      1
62      1
190     1
Name: Glucose, Length: 136, dtype: int64
```

In [8]:
```
plt.hist(df['Glucose'])
```

Out[8]:
```
(array([  5.,   0.,   4.,  32., 156., 211., 163.,  95.,  56.,  46.]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 <BarContainer object of 10 artists>)
```

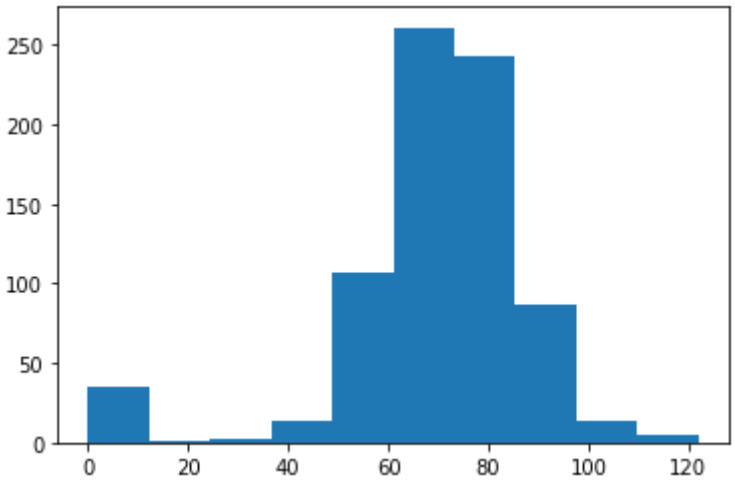


In [9]:
```
df['BloodPressure'].value_counts()
```

Out[9]:
```
70     57
74     52
78     45
68     45
72     44
64     43
80     40
76     39
60     37
0      35
62     34
66     30
82     30
88     25
84     23
90     22
86     21
58     21
50     13
56     12
52     11
54     11
75      8
92      8
65      7
85      6
94      6
48      5
96      4
44      4
100     3
106     3
98      3
110     3
55      2
108     2
104     2
46      2
30      2
122     1
95      1
102     1
61      1
24      1
38      1
40      1
114     1
Name: BloodPressure, dtype: int64
```

In [10]:
```
plt.hist(df['BloodPressure'])
```

Out[10]:
```
(array([ 35.,   1.,   2.,  13., 107., 261., 243.,  87.,  14.,   5.]),
 array([  0. ,  12.2,  24.4,  36.6,  48.8,  61. ,  73.2,  85.4,  97.6,
        109.8, 122. ]),
 <BarContainer object of 10 artists>)
```

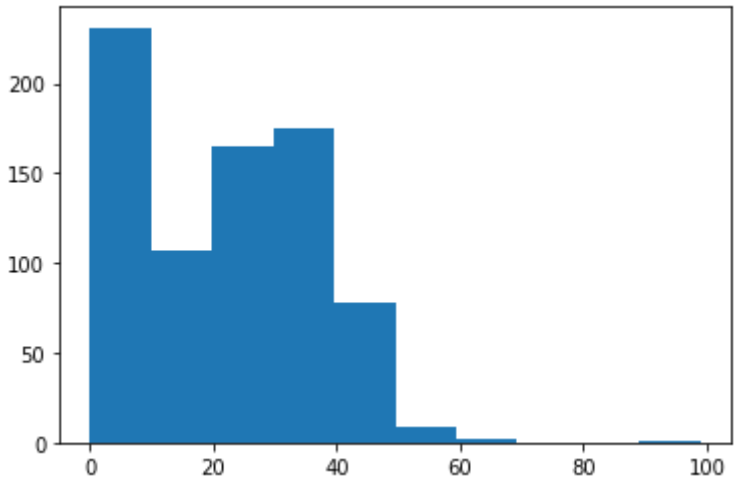


In [11]:
```
df['SkinThickness'].value_counts()
```

Out[11]:
```
0     227
32     31
30     27
27     23
23     22
33     20
28     20
18     20
31     19
19     18
39     18
29     17
40     16
25     16
26     16
22     16
37     16
41     15
35     15
36     14
15     14
17     14
20     13
24     12
42     11
13     11
21     10
46      8
34      8
12      7
38      7
11      6
43      6
16      6
45      6
14      6
44      5
10      5
48      4
47      4
49      3
50      3
8       2
7       2
52      2
54      2
63      1
60      1
56      1
51      1
99      1
Name: SkinThickness, dtype: int64
```

In [12]:
```
plt.hist(df['SkinThickness'])
```

Out[12]:
```
(array([231., 107., 165., 175.,  78.,   9.,   2.,   0.,   0.,   1.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 <BarContainer object of 10 artists>)
```

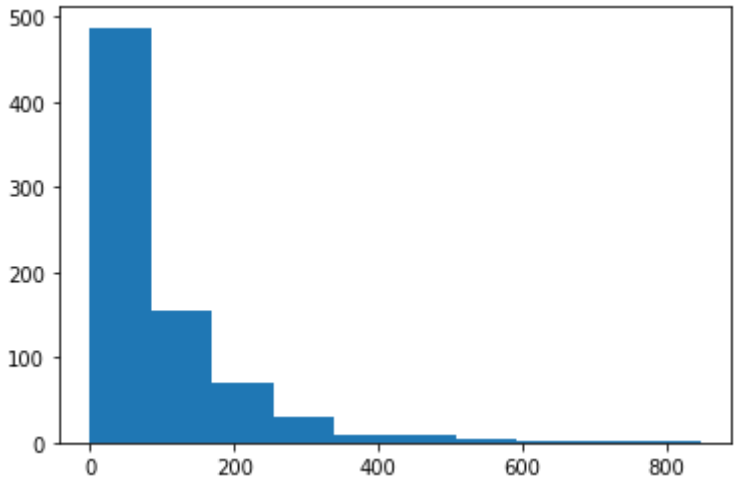


In [13]:
```
df['Insulin'].value_counts()
```

Out[13]:
```
0      374
105     11
130      9
140      9
120      8
      ...
73       1
171      1
255      1
52       1
112      1
Name: Insulin, Length: 186, dtype: int64
```

In [14]:
```
plt.hist(df['Insulin'])
```

Out[14]:
```
(array([487., 155.,  70.,  30.,   8.,   9.,   5.,   1.,   2.,   1.]),
 array([  0. ,  84.6, 169.2, 253.8, 338.4, 423. , 507.6, 592.2, 676.8,
        761.4, 846. ]),
 <BarContainer object of 10 artists>)
```

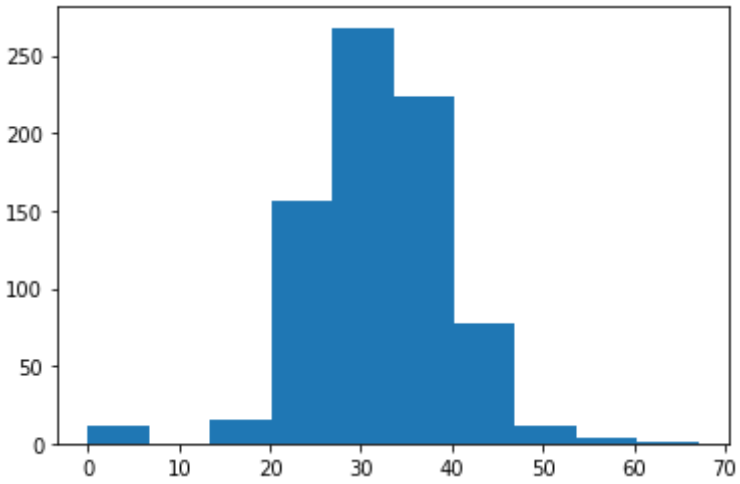


In [15]:
```
df['BMI'].value_counts()
```

Out[15]:
```
32.0    13
31.6    12
31.2    12
0.0     11
32.4    10
        ..
36.7     1
41.8     1
42.6     1
42.8     1
46.3     1
Name: BMI, Length: 248, dtype: int64
```

In [16]:
```
plt.hist(df['BMI'])
```

Out[16]:
```
(array([ 11.,   0.,  15., 156., 268., 224.,  78.,  12.,   3.,   1.]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 <BarContainer object of 10 artists>)
```



In [17]:
```
df.describe()
```

Out[17]:

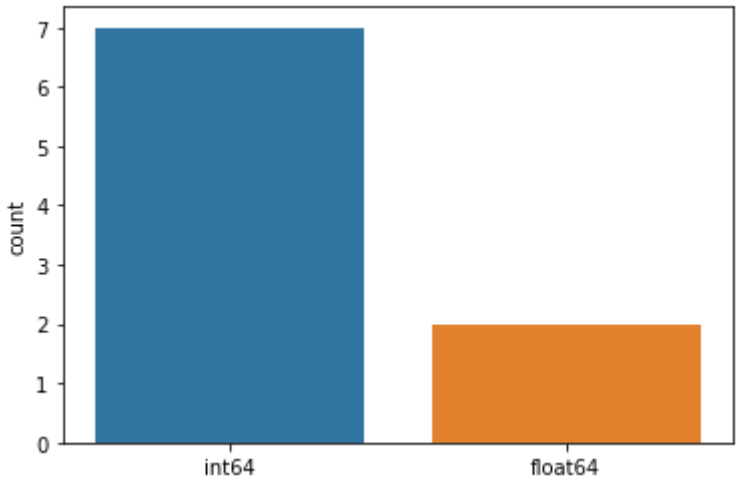
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPe
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	

In [18]:
```
print(df.dtypes.unique())
```

```
[dtype('int64') dtype('float64')]
```

In [19]:
```
data_type_counts = df.dtypes.value_counts()
#Create a count (frequency) plot describing the data types and the count of variabl
sns.countplot(x=df.dtypes.map(str))
```

Out[19]:
```
<AxesSubplot:ylabel='count'>
```

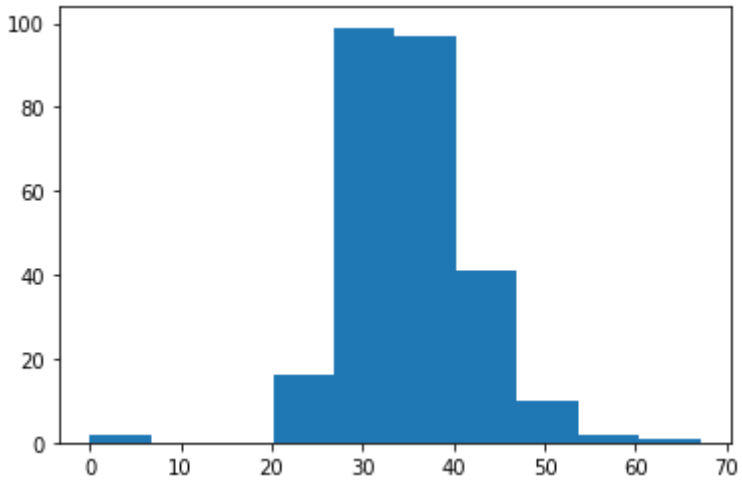


Project Task: Week 2 Data Exploration:

1. Check the balance of the data by plotting the count of outcomes by their value. Describe your findings and plan future course of action.
2. Create scatter charts between the pair of variables to understand the relationships. Describe your findings.
3. Perform correlation analysis. Visually explore it using a heat map.

In [20]:
```
plt.hist(new_df['BMI'])
```

Out[20]:
```
(array([ 2.,  0.,  0., 16., 99., 97., 41., 10.,  2.,  1.]),
 array([ 0.  ,  6.71, 13.42, 20.13, 26.84, 33.55, 40.26, 46.97, 53.68,
        60.39, 67.1 ]),
 <BarContainer object of 10 artists>)
```

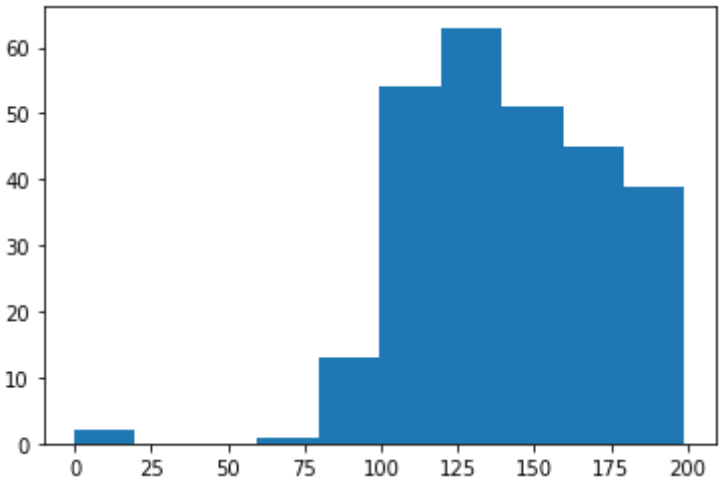


In [21]:
```
new_df['BMI'].value_counts
```

Out[21]:
```
<bound method IndexOpsMixin.value_counts of 0      33.6
2      23.3
4      43.1
6      31.0
8      30.5
       ...
755    36.5
757    36.3
759    35.5
761    44.0
766    30.1
Name: BMI, Length: 268, dtype: float64>
```

In [22]:
```
plt.hist(new_df['Glucose'])
```

Out[22]:
```
(array([ 2.,  0.,  0.,  1., 13., 54., 63., 51., 45., 39.]),
 array([  0. ,  19.9,  39.8,  59.7,  79.6,  99.5, 119.4, 139.3, 159.2,
        179.1, 199. ]),
 <BarContainer object of 10 artists>)
```

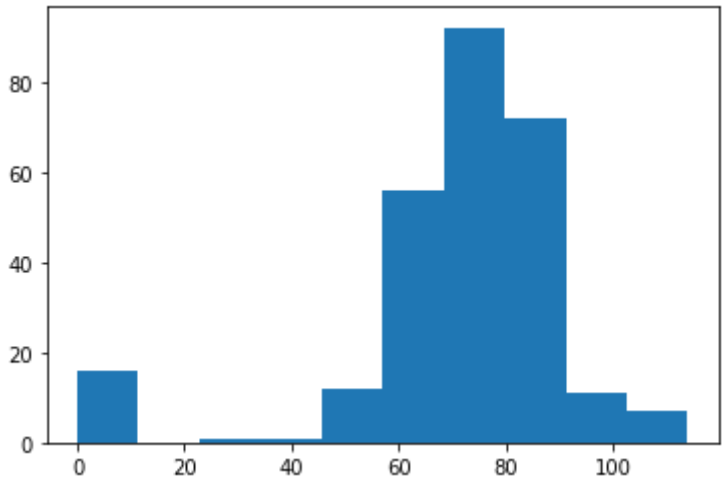


In [23]:
```
new_df['Glucose'].value_counts
```

Out[23]:
```
<bound method IndexOpsMixin.value_counts of 0      148
2      183
4      137
6       78
8      197
      ...
755    128
757    123
759    190
761    170
766    126
Name: Glucose, Length: 268, dtype: int64>
```

In [24]:
```
plt.hist(new_df['BloodPressure'])
```

Out[24]:
```
(array([16.,  0.,  1.,  1., 12., 56., 92., 72., 11.,  7.]),
 array([  0. ,  11.4,  22.8,  34.2,  45.6,  57. ,  68.4,  79.8,  91.2,
        102.6, 114. ]),
 <BarContainer object of 10 artists>)
```

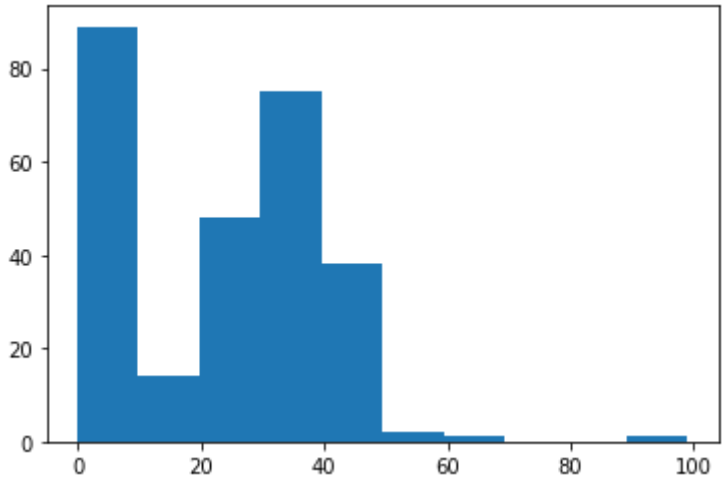


In [25]:
```
new_df['BloodPressure'].value_counts
```

Out[25]:
```
<bound method IndexOpsMixin.value_counts of 0      72
2      64
4      40
6      50
8      70
       ..
755    88
757    72
759    92
761    74
766    60
Name: BloodPressure, Length: 268, dtype: int64>
```

In [26]:
```
plt.hist(new_df['SkinThickness'])
```

Out[26]:
```
(array([89., 14., 48., 75., 38.,  2.,  1.,  0.,  0.,  1.]),
 array([ 0. ,  9.9, 19.8, 29.7, 39.6, 49.5, 59.4, 69.3, 79.2, 89.1, 99. ]),
 <BarContainer object of 10 artists>)
```

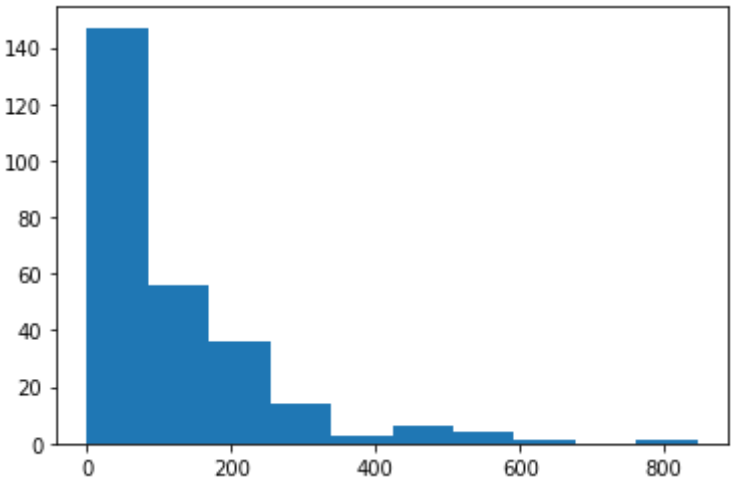


In [27]:
```
new_df['SkinThickness'].value_counts
```

Out[27]:
```
<bound method IndexOpsMixin.value_counts of 0      35
2       0
4      35
6      32
8      45
       ..
755    39
757     0
759     0
761    31
766     0
Name: SkinThickness, Length: 268, dtype: int64>
```

In [28]:
```
plt.hist(new_df['Insulin'])
```

Out[28]:
```
(array([147.,  56.,  36.,  14.,   3.,   6.,   4.,   1.,   0.,   1.]),
 array([  0. ,  84.6, 169.2, 253.8, 338.4, 423. , 507.6, 592.2, 676.8,
        761.4, 846. ]),
 <BarContainer object of 10 artists>)
```

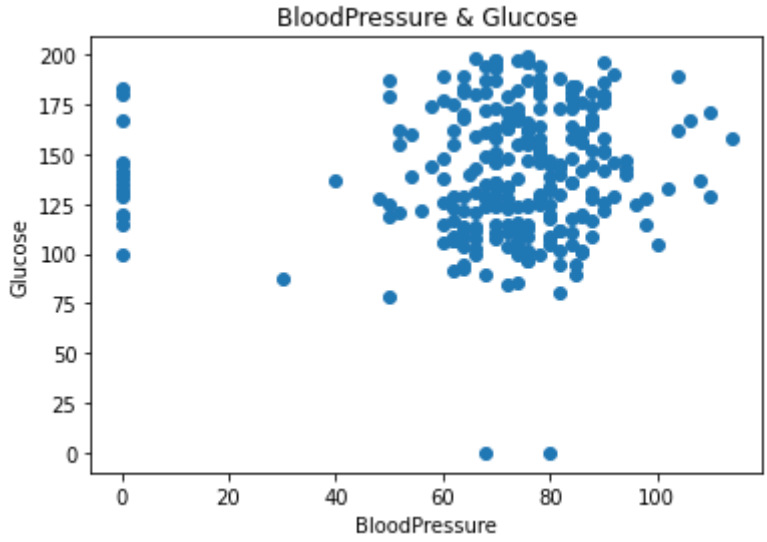


In [29]:
```
new_df['Insulin'].value_counts
```

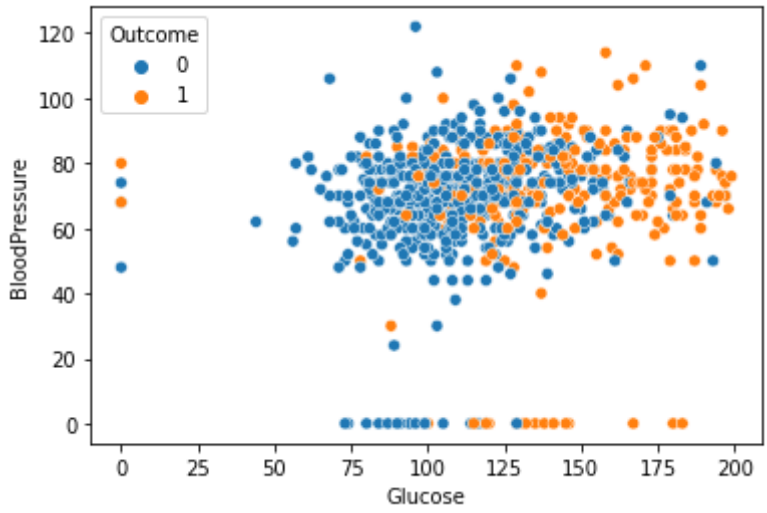
Out[29]:
```
<bound method IndexOpsMixin.value_counts of 0        0
2        0
4      168
6       88
8      543
      ...
755    110
757      0
759      0
761      0
766      0
Name: Insulin, Length: 268, dtype: int64>
```

In [30]:
```
BP = new_df['BloodPressure']
Glucose = new_df['Glucose']
SkinThickness = new_df['SkinThickness']
Insulin = new_df['Insulin']
BMI = new_df['BMI']
```

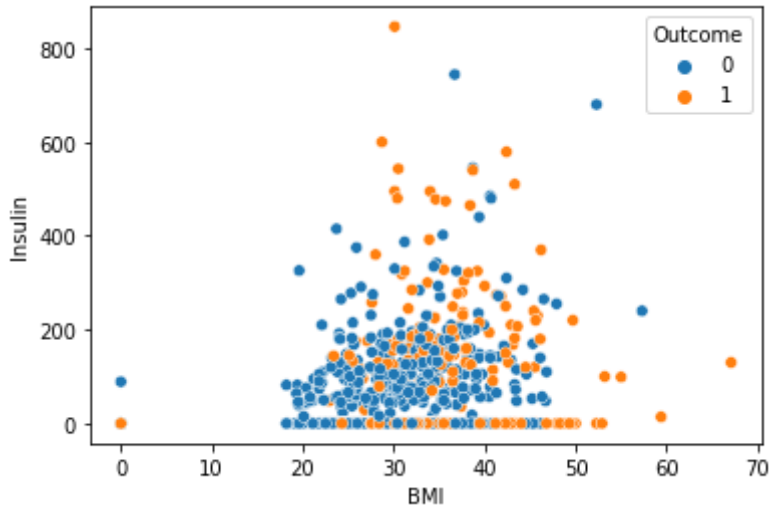
In [31]:
```
plt.scatter(BP, Glucose)
plt.xlabel('BloodPressure')
plt.ylabel('Glucose')
plt.title('BloodPressure & Glucose')
plt.show()
```



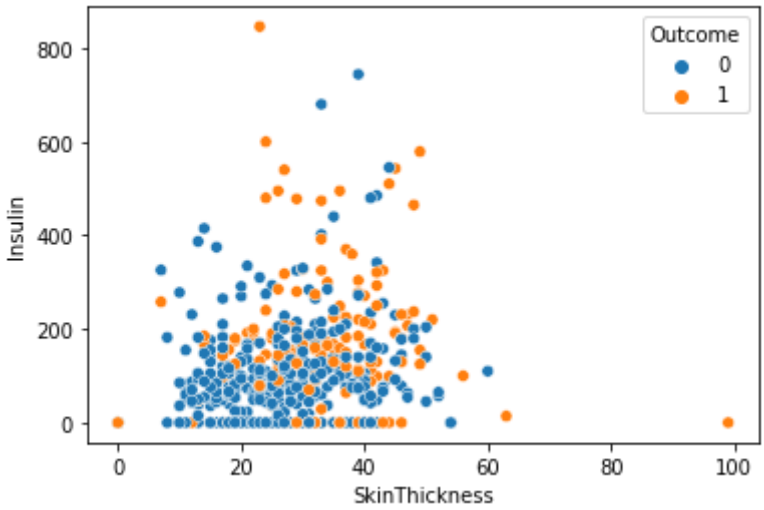
In [32]:
```
sns.scatterplot(x= "Glucose" ,y= "BloodPressure",
                hue="Outcome",
                data=df);
```



In [33]:
```
sns.scatterplot(x= "BMI" ,y= "Insulin",
                hue="Outcome",
                data=df);
```



In [34]:
```
sns.scatterplot(x= "SkinThickness" ,y= "Insulin",
                hue="Outcome",
                data=df);
```



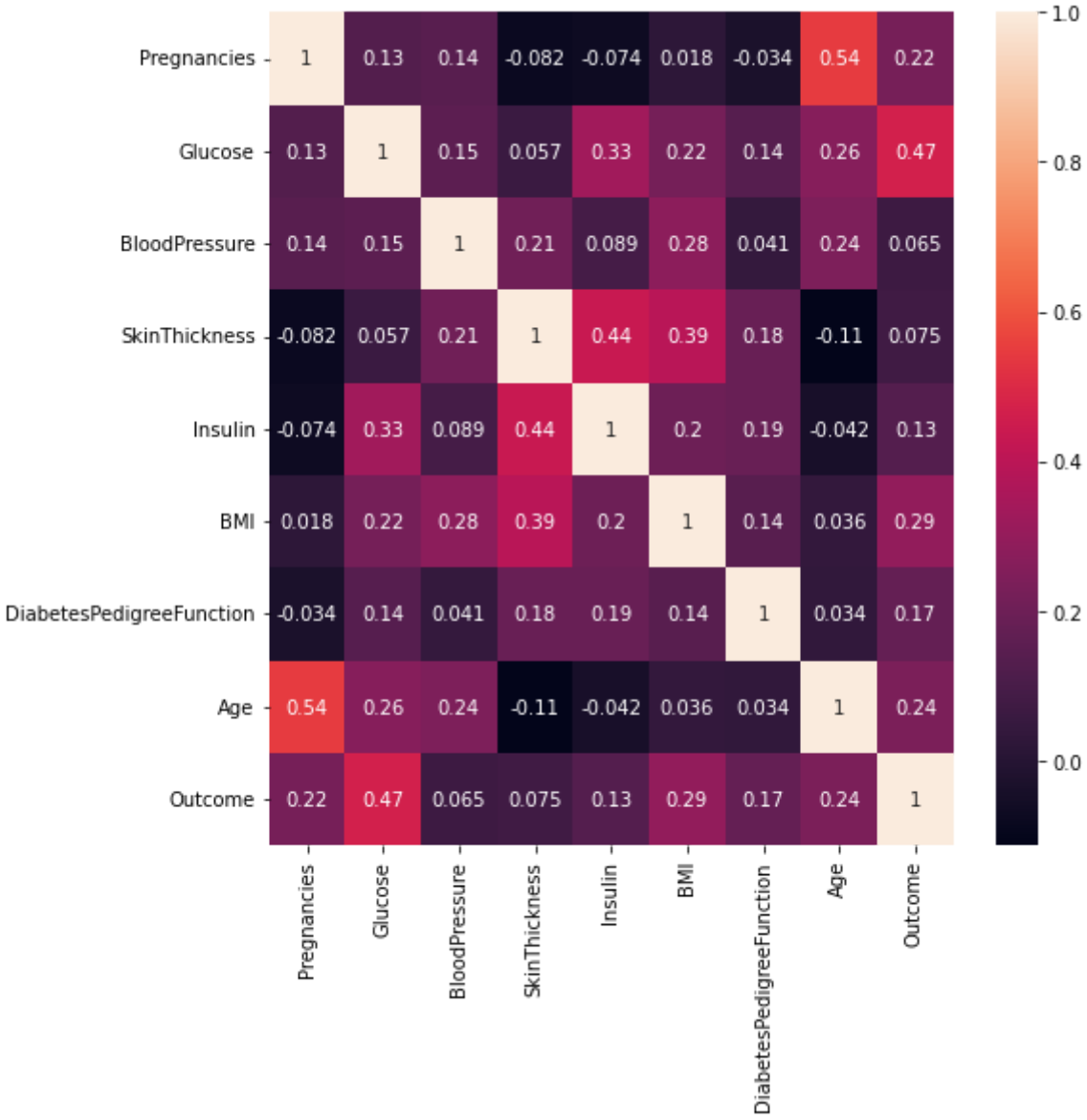
In [35]:
```
df.corr()
```

Out[35]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BM
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.01768
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.22107
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.28180
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.39257
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.19785
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.00000
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.14064
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.03624
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.29269

In [36]:
```
plt.subplots(figsize=(8,8))
sns.heatmap(df.corr(),annot=True)
```

Out[36]:
```
<AxesSubplot:>
```



Project Task: Week 3 Data Modeling:

1. Devise strategies for model building. It is important to decide the right validation framework. Express your thought process.
2. Apply an appropriate classification algorithm to build a model. Compare various models with the results from KNN algorithm.

In [37]:
```
#Train test split
from sklearn.model_selection import train_test_split
```

In [38]:
```
X=df.drop(['DiabetesPedigreeFunction'],axis=1)
y=df['DiabetesPedigreeFunction']
y=y.astype('str')
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [50]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X_train,y_train)
```

Out[50]:
```
LogisticRegression()
```

In [51]:
```
print(model.score(X_train,y_train))
print(model.score(X_test,y_test))
```

```
0.7719869706840391
0.7662337662337663
```

In [52]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(label,model.predict(features))
cm
```

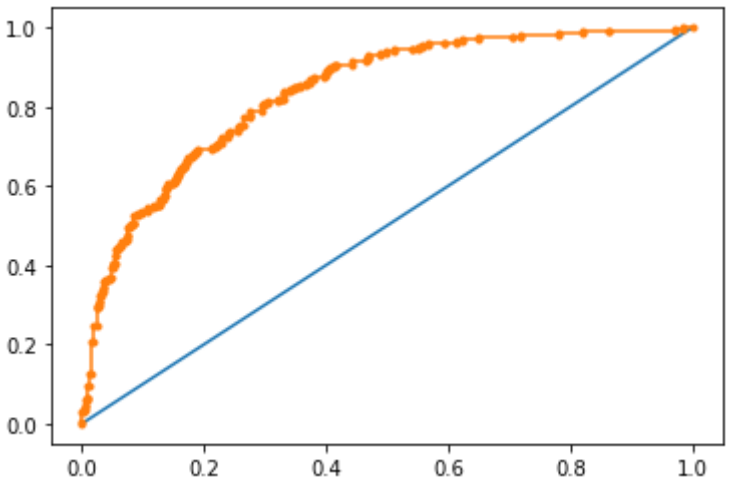
Out[52]:
```
array([[446,  54],
       [122, 146]], dtype=int64)
```

In [53]:
```
from sklearn.metrics import classification_report
print(classification_report(label,model.predict(features)))
```

```
              precision    recall  f1-score   support

           0       0.79      0.89      0.84       500
           1       0.73      0.54      0.62       268

    accuracy                           0.77       768
   macro avg       0.76      0.72      0.73       768
weighted avg       0.77      0.77      0.76       768
```

In [59]:
```
#ROC Curve
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score


prob = model.predict_proba(features)
prob = prob[:, 1]
auc = roc_auc_score(label, prob)
print('AUC: %.3f' % auc)
fpr, tpr, thresholds = roc_curve(label, prob)
plt.plot([0, 1], [0, 1])
plt.plot(fpr, tpr, marker='.')
```

Out[59]:
```
AUC: 0.837
[<matplotlib.lines.Line2D at 0x16f14e0f3a0>]
```



In [60]:
```
#Decission Tree Classifier
from sklearn.tree import DecisionTreeClassifier
model3 = DecisionTreeClassifier(max_depth=5)
model3.fit(X_train,y_train)
```

Out[60]:
```
DecisionTreeClassifier(max_depth=5)
```

In [62]:
```
model3.score(X_train,y_train)
```

Out[62]:
```
0.8289902280130294
```

In [63]:
```
model3.score(X_test,y_test)
```

Out[63]:
```
0.7727272727272727
```

In [64]:
```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
model4 = RandomForestClassifier(n_estimators=11)
model4.fit(X_train,y_train)
```

Out[64]:
```
RandomForestClassifier(n_estimators=11)
```

In [65]:
```
model4.score(X_train,y_train)
```

Out[65]:
```
0.9869706840390879
```

In [66]:
```
model4.score(X_test,y_test)
```

Out[66]:
```
0.7012987012987013
```

In [67]:
```
#Support Vector Classifier

from sklearn.svm import SVC
model5 = SVC(kernel='rbf',gamma='auto')
model5.fit(X_train,y_train)
```

Out[67]:
```
SVC(gamma='auto')
```

In [70]:
```
model5.score(X_test,y_test)
```

Out[70]:
```
0.6168831168831169
```

In [71]:
```
model5.score(X_train,y_train)
```

Out[71]:
```
1.0
```

In [72]:
```
#K-NN
from sklearn.neighbors import KNeighborsClassifier
model2 = KNeighborsClassifier(n_neighbors=7,
                              metric='minkowski',
                              p = 2)
model2.fit(X_train,y_train)
```

Out[72]:
```
KNeighborsClassifier(n_neighbors=7)
```

In [ ]:
```

```