In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
```

In [3]:
```
import os
os.chdir(r"C:\Users\varshini rajkumar\Desktop\python")
df=pd.read_csv("diabetes.csv")
```

In [6]:
```
df.head(77).T
```

Out[6]:

	0	1	2	3	4	5	6	7	8	9	...	67	68	69	
Pregnancies	6.000	1.000	8.000	1.000	0.000	5.000	3.000	10.000	2.000	8.000	...	2.000	1.000	4.000	2.0
Glucose	148.000	85.000	183.000	89.000	137.000	116.000	78.000	115.000	197.000	125.000	...	109.000	95.000	146.000	100.0
BloodPressure	72.000	66.000	64.000	66.000	40.000	74.000	50.000	0.000	70.000	96.000	...	92.000	66.000	85.000	66.0
SkinThickness	35.000	29.000	0.000	23.000	35.000	0.000	32.000	0.000	45.000	0.000	...	0.000	13.000	27.000	20.0
Insulin	0.000	0.000	0.000	94.000	168.000	0.000	88.000	0.000	543.000	0.000	...	0.000	38.000	100.000	90.0
BMI	33.600	26.600	23.300	28.100	43.100	25.600	31.000	35.300	30.500	0.000	...	42.700	19.600	28.900	32.9
DiabetesPedigreeFunction	0.627	0.351	0.672	0.167	2.288	0.201	0.248	0.134	0.158	0.232	...	0.845	0.334	0.189	0.8
Age	50.000	31.000	32.000	21.000	33.000	30.000	26.000	29.000	53.000	54.000	...	54.000	25.000	27.000	28.0
Outcome	1.000	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000	1.000	...	0.000	0.000	0.000	1.0

9 rows × 77 columns

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

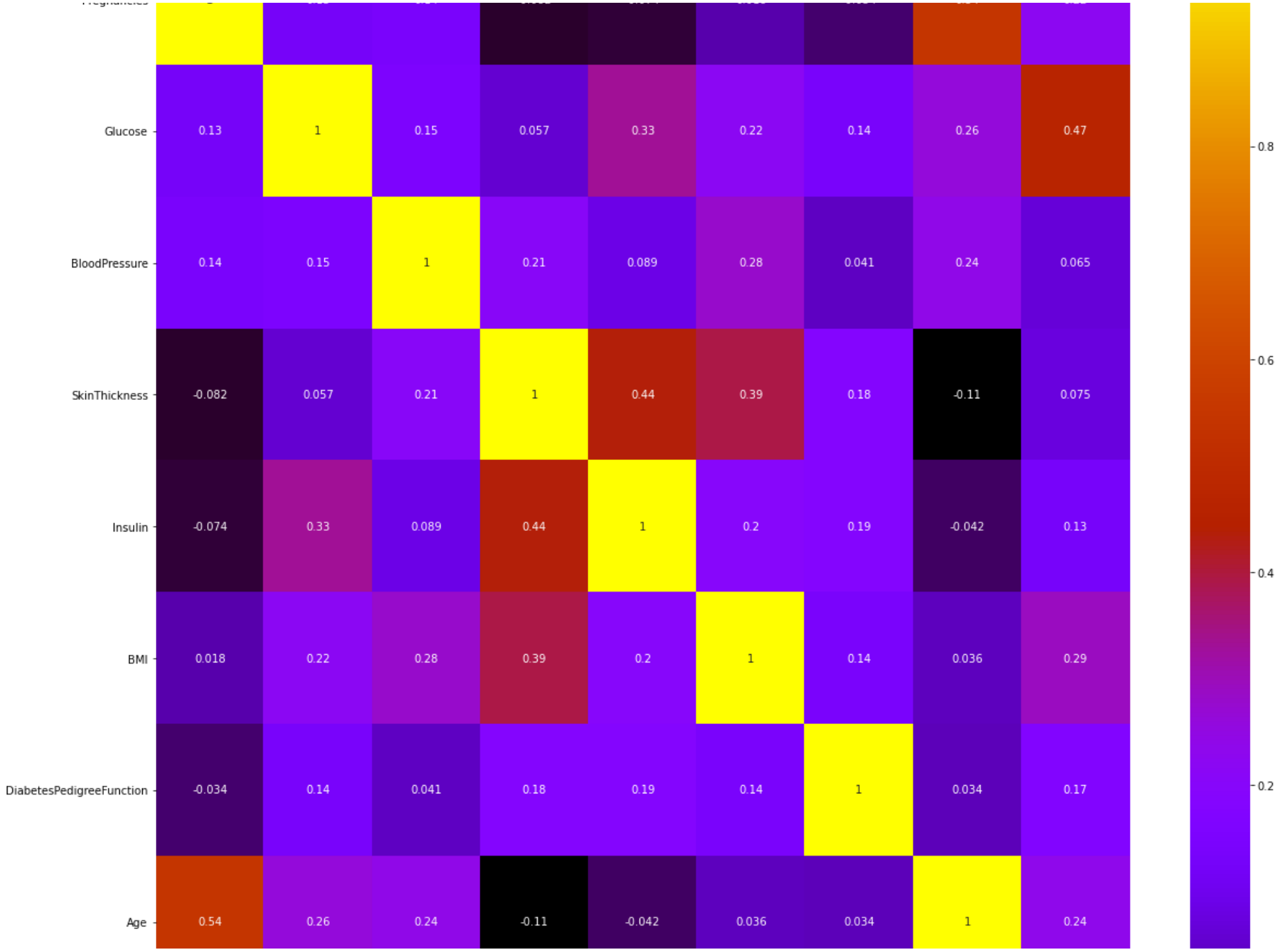
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   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
```

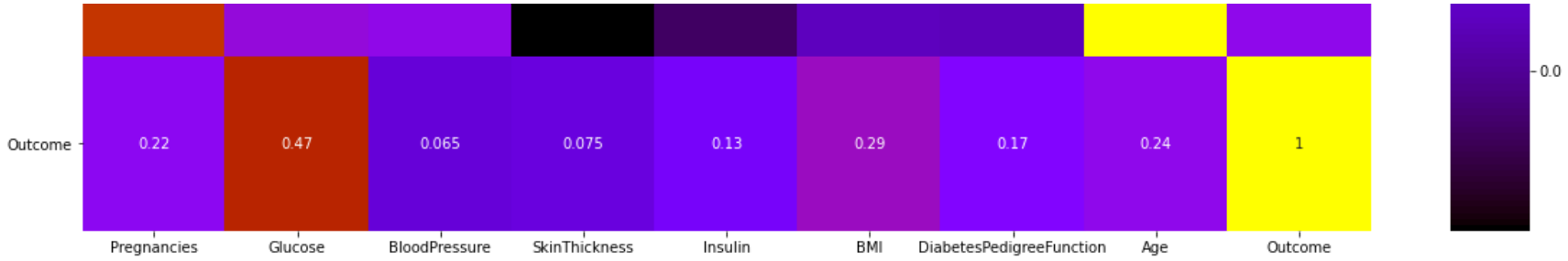
In [7]:
```
df.describe()
```

Out[7]:

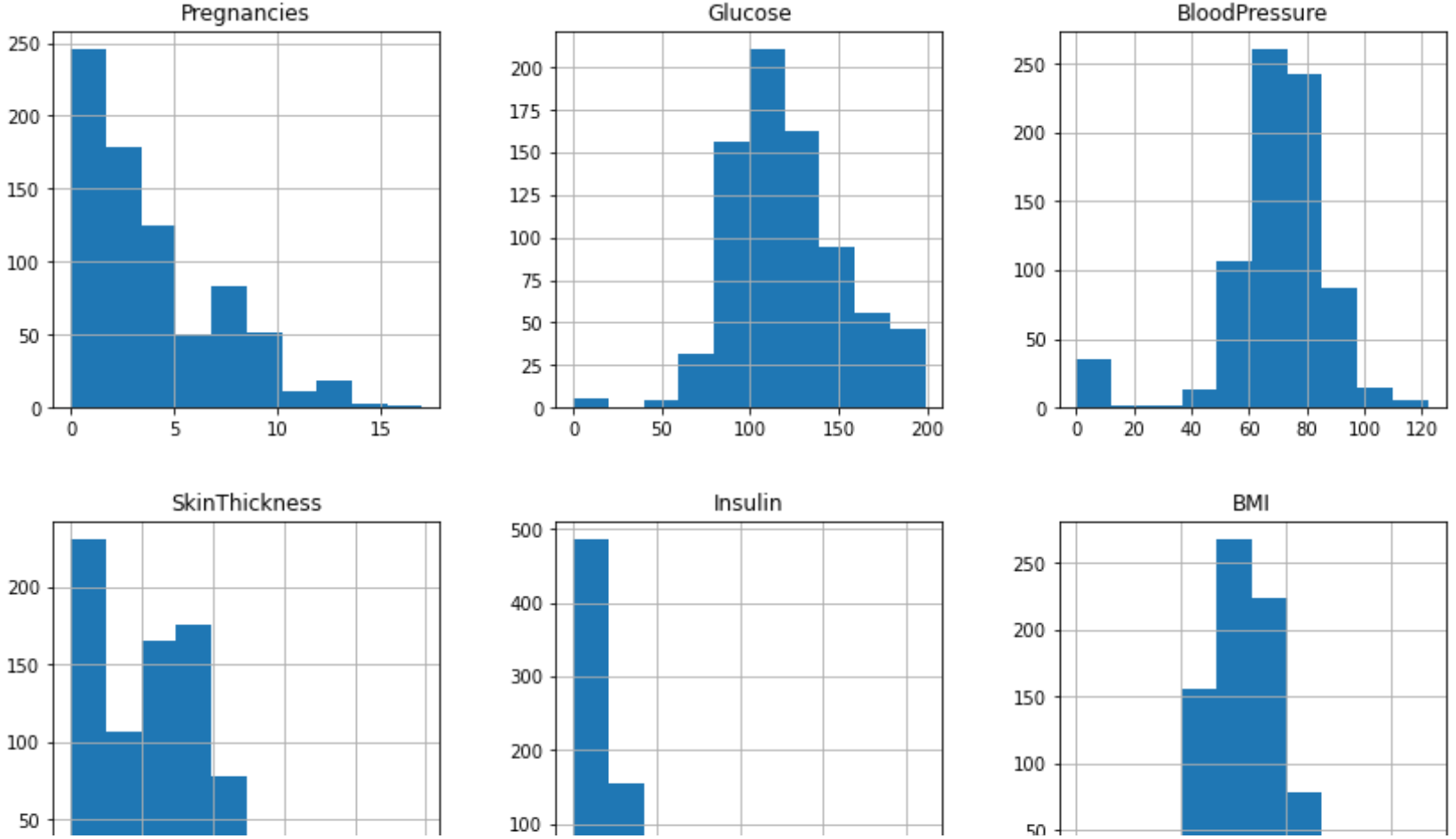
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

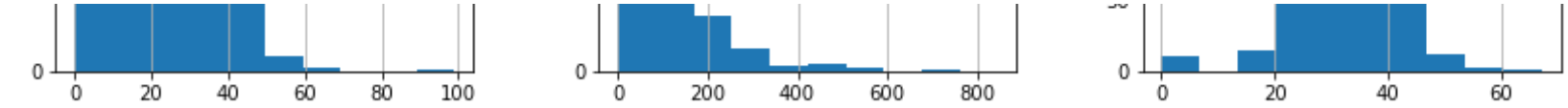
In [8]:
```
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="gnuplot")
```

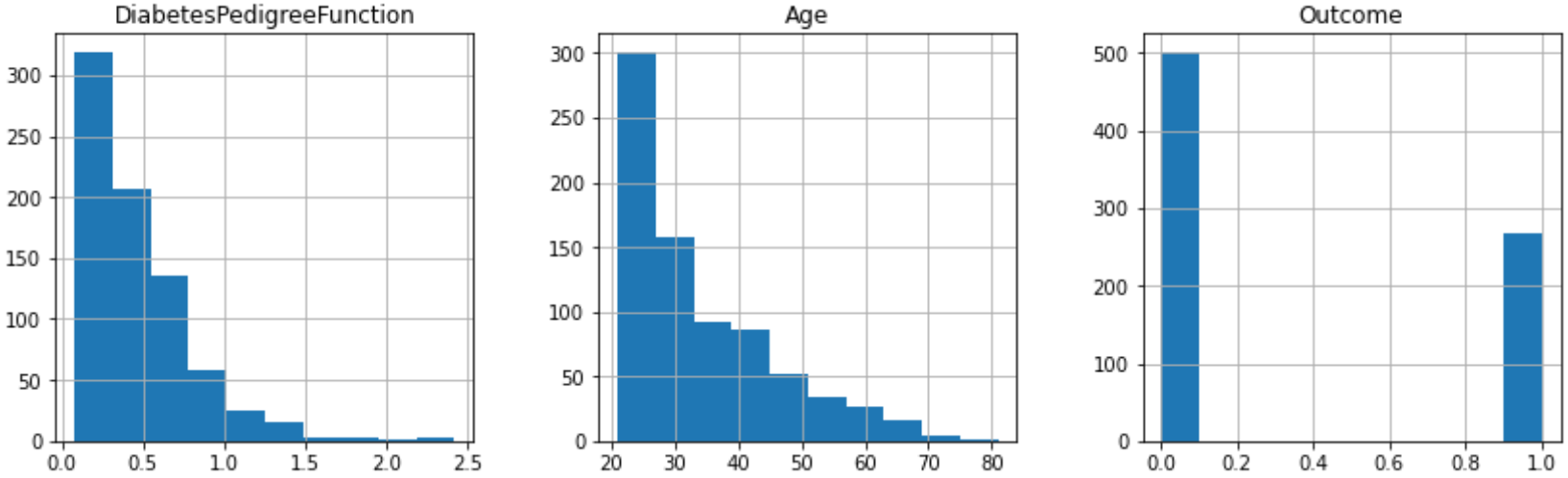




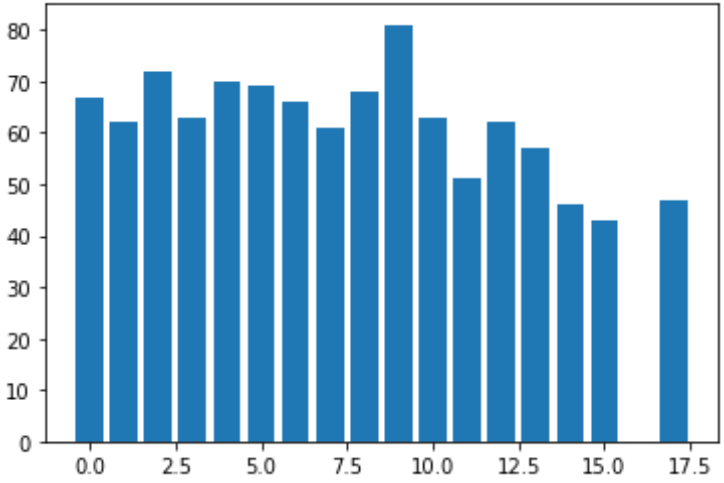
In [9]:
```
df.hist(figsize=(14,14))
plt.show()
```





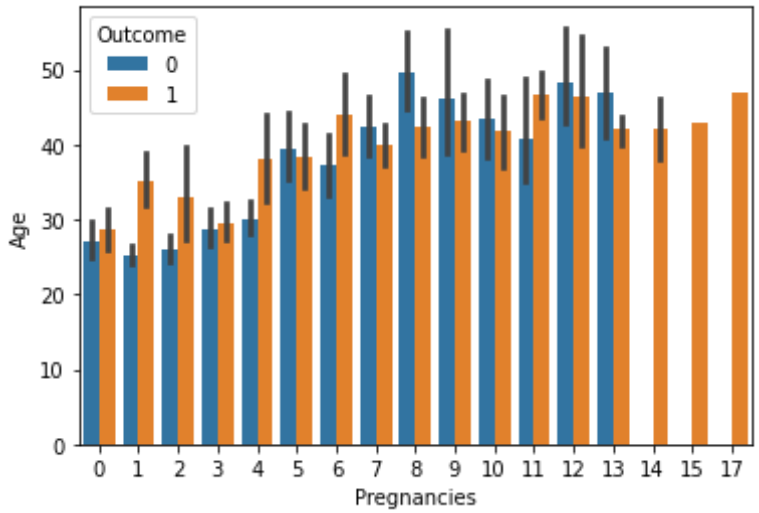


In [11]:
```
plt.bar(x=df['Pregnancies'],height=df['Age'])
plt.show()
```



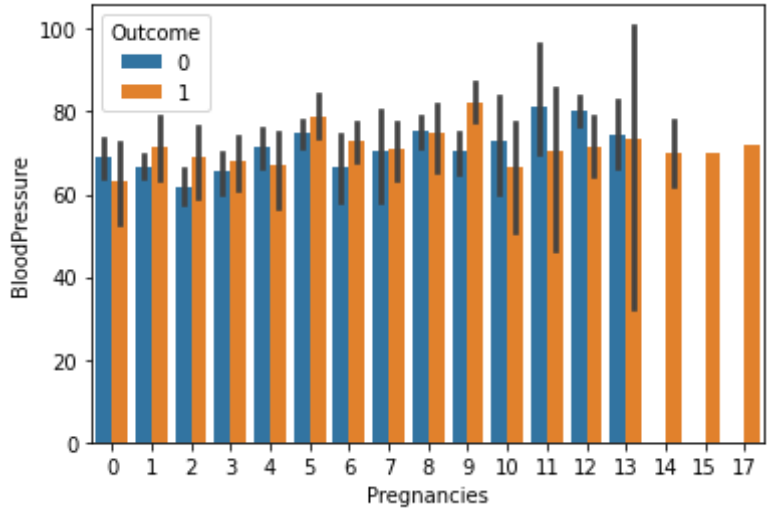
In [13]:
```
sns.barplot(x=df['Pregnancies'],y=df['Age'],hue=df['Outcome'])
```

Out[13]:
```
<AxesSubplot:xlabel='Pregnancies', ylabel='Age'>
```



In [14]:
```
sns.barplot(x=df['Pregnancies'],y=df['BloodPressure'],hue=df['Outcome'])
```

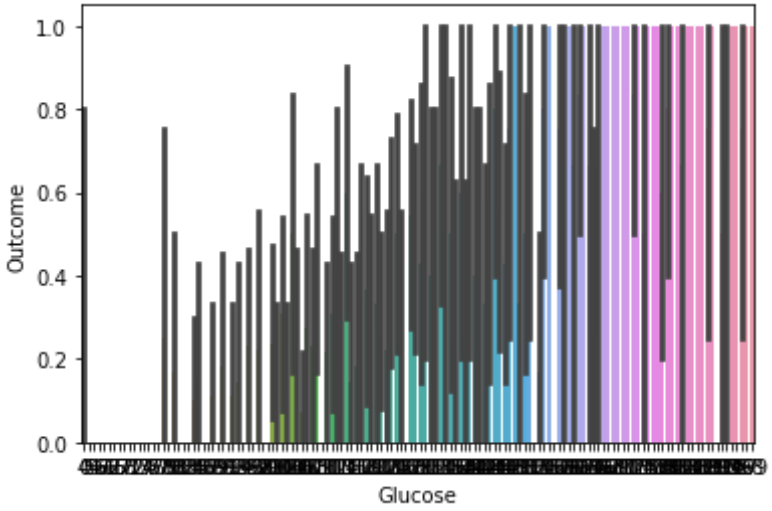
Out[14]:
```
<AxesSubplot:xlabel='Pregnancies', ylabel='BloodPressure'>
```



In [16]:
```
import warnings
warnings.filterwarnings('ignore')
```

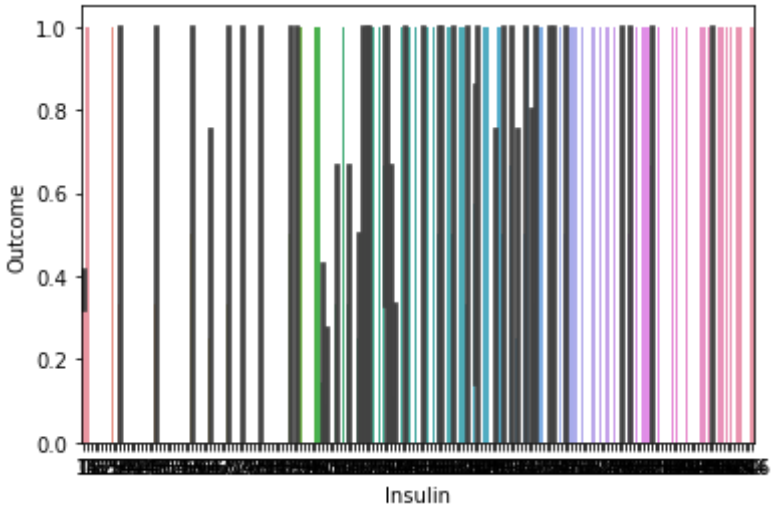
In [17]:
```
sns.barplot(df["Glucose"],df['Outcome'])
```

Out[17]:
```
<AxesSubplot:xlabel='Glucose', ylabel='Outcome'>
```



In [18]:
```
sns.barplot(df["Insulin"],df['Outcome'])
```

Out[18]:
```
<AxesSubplot:xlabel='Insulin', ylabel='Outcome'>
```



In [19]:
```
a.all=['Insulin','Age','Glucose','Pregnancies','DiabetesPedigreeFunction','SkinThickness','DiabetesPedigreeFunction']
```

In [23]:
```
zero_not_accepted=['Glucose','BloodPressure','SkinThickness','Insulin','BMI']
for column in zero_not_accepted:
    df[column]=df[column].replace(0,np.NaN)
    mean=int(df[column].mean(skipna=True))
    df[column]=df[column].replace(np.NaN,mean)
```

In [24]:
```
df['Glucose']
```

Out[24]:
```
0      148.0
1       85.0
2      183.0
3       89.0
4      137.0
       ...
763    101.0
764    122.0
765    121.0
766    126.0
767     93.0
Name: Glucose, Length: 768, dtype: float64
```

In [25]:
```
X=df.iloc[:,0:8]
```

```
Y=df.iloc[:,8]
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=0)
```

In [26]:
```
sc_X=StandardScaler()
X_train=sc_X.fit_transform(X_train)
X_test=sc_X.transform(X_test)
```

In [27]:
```
len(Y)
```

Out[27]:
```
768
```

In [28]:
```
import math
math.sqrt(len(Y_train))
```

Out[28]:
```
24.779023386727733
```

In [29]:
```
math.sqrt(len(Y_test))
```

Out[29]:
```
12.409673645990857
```

In [40]:
```
classifier=KNeighborsClassifier(n_neighbors=11,p=2,metric='euclidean')
classifier.fit(X_train,Y_train)
```

Out[40]:
```
KNeighborsClassifier(metric='euclidean', n_neighbors=11)
```

In [41]:
```
y_pred=classifier.predict(X_test)
y_pred
```

Out[41]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
      dtype=int64)
```

In [42]:
```
cm=confusion_matrix(Y_test,y_pred)
print(cm)
```

```
[[94 13]
 [15 32]]
```

In [43]:
```
print(f1_score(Y_test,y_pred))
```

```
0.6956521739130436
```

In [44]:
```
#knn algorithm accuracy
print((accuracy_score(Y_test,y_pred))*100)
```

```
81.81818181818183
```

In [71]:
```
from sklearn.linear_model import LogisticRegression
LR=LogisticRegression(max_iter=150)
LR.fit(X_train,Y_train)
LogisticRegression(max_iter=150)
```

Out[71]:
```
LogisticRegression(max_iter=150)
```

In [72]:
```
y_pred = LR.predict(X_test)
y_pred
```

Out[72]:
```
array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1,
       1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
      dtype=int64)
```

In [73]:
```
from sklearn.metrics import accuracy_score
print(accuracy_score(y_pred,Y_test))
```

```
0.8116883116883117
```

In [74]:
```
from sklearn import tree
clf=tree.DecisionTreeClassifier(max_depth=10)
clf.fit(X_train,Y_train)
(accuracy_score(y_pred,Y_test)*100)
```

Out[74]:
```
81.16883116883116
```

In [75]:
```
from sklearn.neighbors import KNeighborsClassifier
KN=KNeighborsClassifier(n_neighbors=8)
KN.fit(X_train,Y_train)
y_pred=KN.predict(X_test)
print(accuracy_score(y_pred,Y_test)*100)
```

```
81.81818181818183
```

In [76]:
```
from sklearn import svm
clf=svm.SVC(kernel='rbf')
clf.fit(X_train,Y_train)
y_pred=clf.predict(X_test)
accuracy_score(y_pred,Y_test)*100
```

Out[76]:
```
77.27272727272727
```

In [77]:
```
from sklearn import svm
clf=svm.SVC(kernel='linear')
clf.fit(X_train,Y_train)
y_pred=clf.predict(X_test)
accuracy_score(y_pred,Y_test)*100
```

Out[77]:
```
79.87012987012987
```

In [79]:
```
from sklearn import ensemble
clf=ensemble.GradientBoostingClassifier()
clf.fit(X_train,Y_train)
clf.fit(X_test,Y_test)
(accuracy_score(y_pred,Y_test)*100)
```

Out[79]:
```
79.87012987012987
```

In [ ]: