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

In [3]:
```
df = pd.read_csv("diabetes.csv")
```

In [4]:
```
df.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [5]:
```
df.shape
```

Out[5]:
```
(768, 9)
```

In [6]:
```
df.describe()
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Pedigree	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 [7]:
```
#replace zeros
zero_not_accepted=["Glucose","BloodPressure","SkinThickness","BMI","Insulin"]
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 [8]:
```
df["Glucose"]
```

Out[8]:
```
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 [9]:
```
#split dataset
X=df.iloc[:,0:8]
y=df.iloc[:,8]
```

In [12]:
```
print(X)
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6    148.0           72.0           35.0    155.0  33.6
1              1     85.0           66.0           29.0    155.0  26.6
2              8    183.0           64.0           29.0    155.0  23.3
3              1     89.0           66.0           23.0     94.0  28.1
4              0    137.0           40.0           35.0    168.0  43.1
..           ...      ...            ...            ...      ...   ...
763           10    101.0           76.0           48.0    180.0  32.9
764            2    122.0           70.0           27.0    155.0  36.8
765            5    121.0           72.0           23.0    112.0  26.2
766            1    126.0           60.0           29.0    155.0  30.1
767            1     93.0           70.0           31.0    155.0  30.4

     Pedigree  Age
0       0.627   50
1       0.351   31
2       0.672   32
3       0.167   21
4       2.288   33
..        ...  ...
763     0.171   63
764     0.340   27
765     0.245   30
766     0.349   47
767     0.315   23

[768 rows x 8 columns]
```

In [13]:
```
print(y)
```

```
0      1
1      0
2      1
3      0
4      1
      ..
763    0
764    0
765    0
766    1
767    0
Name: Outcome, Length: 768, dtype: int64
```

In [14]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=0,test_size=0.2)
```

In [15]:
```
#feature Scaling
sc_X=StandardScaler()
X_train=sc_X.fit_transform(X_train)

X_test=sc_X.transform(X_test)
```

In [16]:
```
knn=KNeighborsClassifier(n_neighbors=11)
```

In [17]:
```
knn.fit(X_train,y_train)
```

Out[17]:
```
KNeighborsClassifier(n_neighbors=11)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

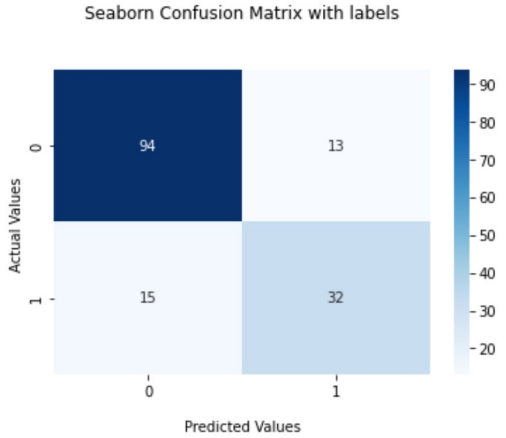
In [18]:
```
y_pred=knn.predict(X_test)
```

In [19]:
```
#Evaluate The Model
cf_matrix=confusion_matrix(y_test,y_pred)
```

In [20]:
```
ax = sns.heatmap(cf_matrix, annot=True, cmap='Blues')

ax.set_title('Seaborn Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');


## Display the visualization of the Confusion Matrix.
plt.show()
```



In [21]:
```
tn, fp, fn, tp = confusion_matrix(y_test, y_pred ).ravel()
tn, fp, fn, tp
```

Out[21]: (94, 13, 15, 32)

In [22]:
```
#The accuracy rate is equal to (tn+tp)/(tn+tp+fn+fp)
accuracy_score(y_test,y_pred)
```

Out[22]: 0.8181818181818182

In [23]:
```
#The precision is the ratio of tp/(tp + fp)
precision_score(y_test,y_pred)
```

Out[23]: 0.7111111111111111

In [24]:
```
##The recall is the ratio of tp/(tp + fn)
recall_score(y_test,y_pred)
```

Out[24]: 0.6808510638297872

In [25]:
```
#error rate=1-accuracy which is lies bertween 0 and 1
error_rate=1-accuracy_score(y_test,y_pred)
```

In [26]:
```
error_rate
```

Out[26]: 0.18181818181818177