Importing Libararies

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
from sklearn import metrics
```

Retreving Data

In [2]:
```
data=pd.read_csv(r"C:\DsTraining\five dataset for clening\diabetes.csv")
data.head(10)
```

Out[2]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	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
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Data Cleaning

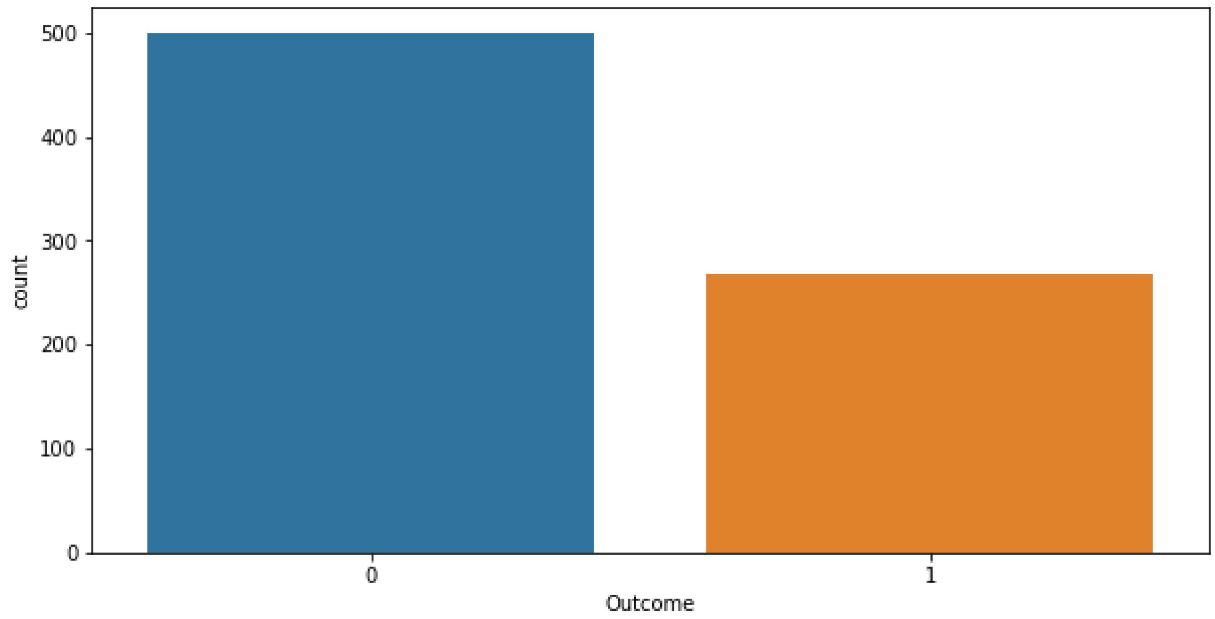
In [3]:
```
#find missing values and fill using appropriate central tendencies.(no missing values find in this data set
data.isnull().sum()
```

Out[3]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

EDA

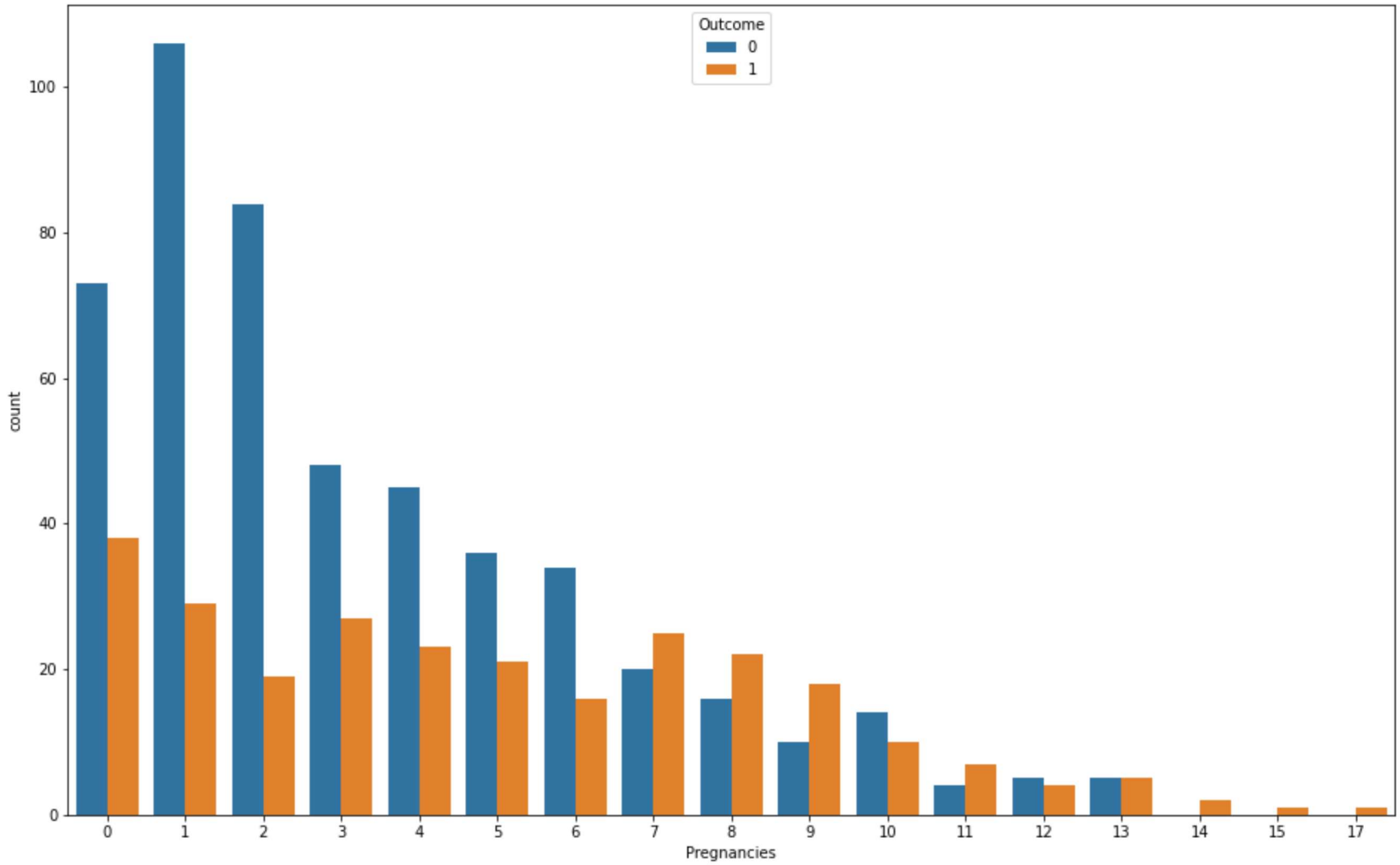
In [4]:
```
plt.figure(figsize=(10,5))
sns.countplot(data["Outcome"])
```

Out[4]:
```
<AxesSubplot:xlabel='Outcome', ylabel='count'>
```



In [5]:
```
plt.figure(figsize=(16,10))
sns.countplot(data["Pregnancies"],hue="Outcome",data=data)
```

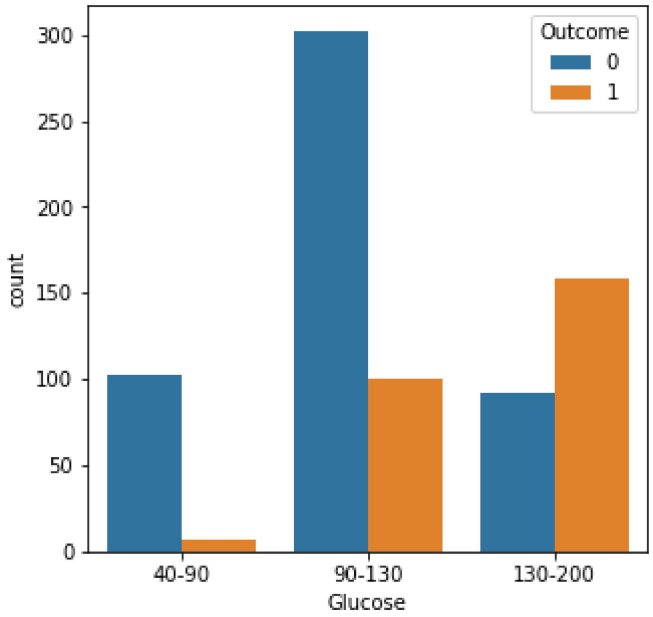
Out[5]: <AxesSubplot:xlabel='Pregnancies', ylabel='count'>



In [6]:
```
glucose_bins=pd.cut(data["Glucose"],bins=[40,90,130,200],labels=["40-90","90-130","130-200"])
```

In [7]:
```
plt.figure(figsize=(5,5))
sns.countplot(glucose_bins,hue=data["Outcome"])
```

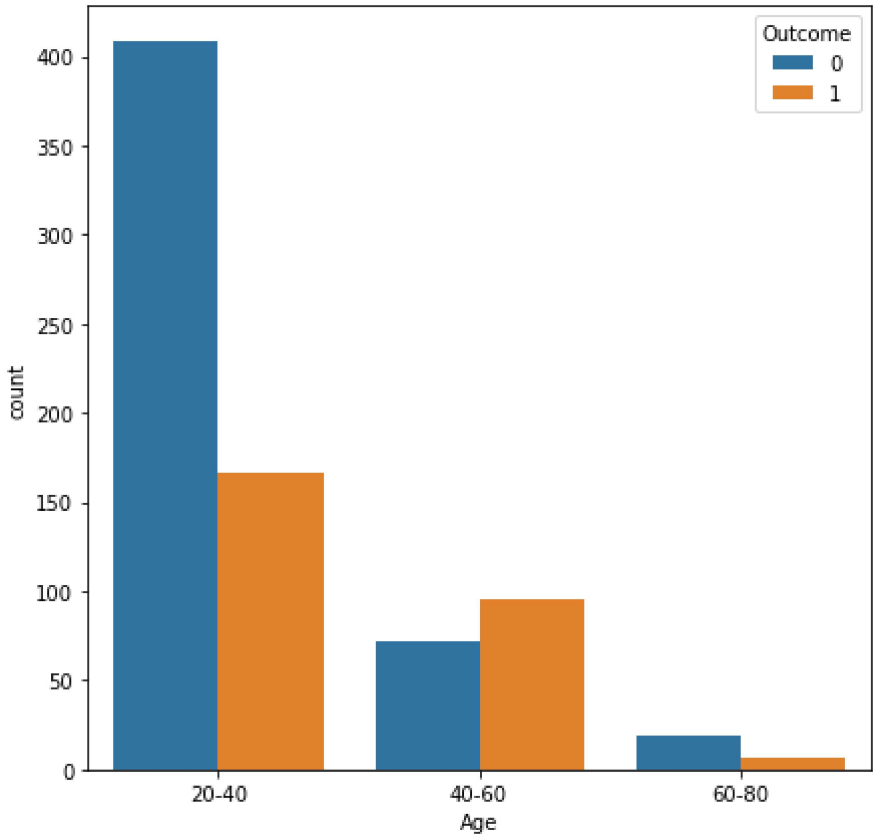
Out[7]: <AxesSubplot:xlabel='Glucose', ylabel='count'>



In [8]:
```
age_bins=pd.cut(data["Age"],bins=[20,40,60,80],labels=["20-40","40-60","60-80"])
```

In [9]:
```
plt.figure(figsize=(7,7))
sns.countplot(age_bins,hue="Outcome",data=data)
```

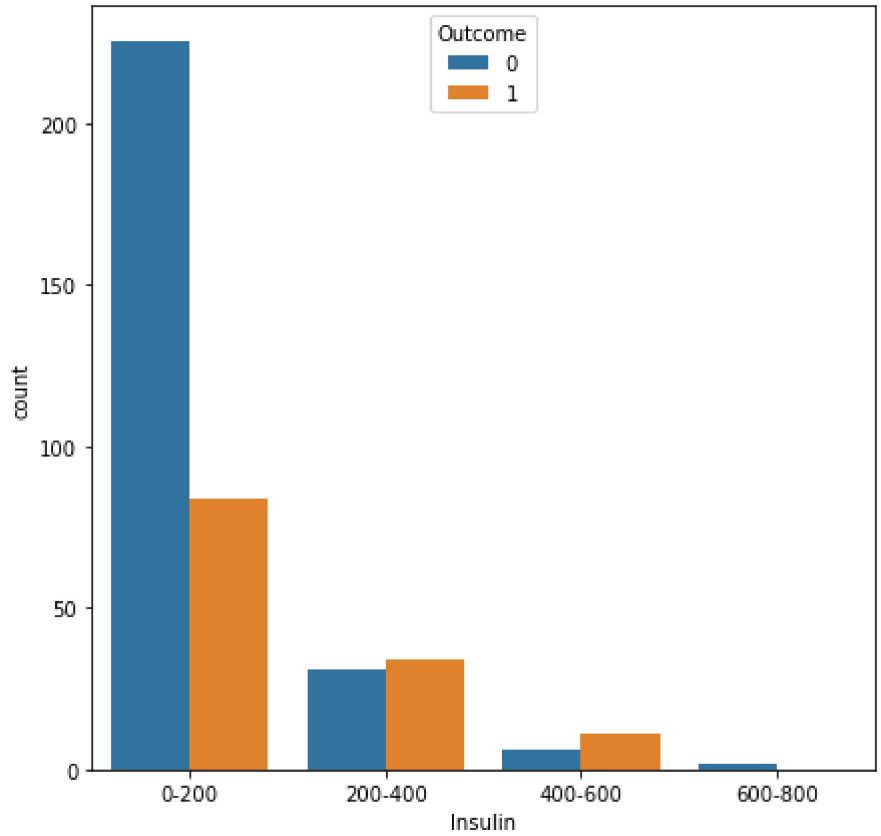
Out[9]: <AxesSubplot:xlabel='Age', ylabel='count'>



In [10]:
```
insulin_bins=pd.cut(data["Insulin"],bins=[0,200,400,600,800],labels=["0-200","200-400","400-600","600-800"]
```

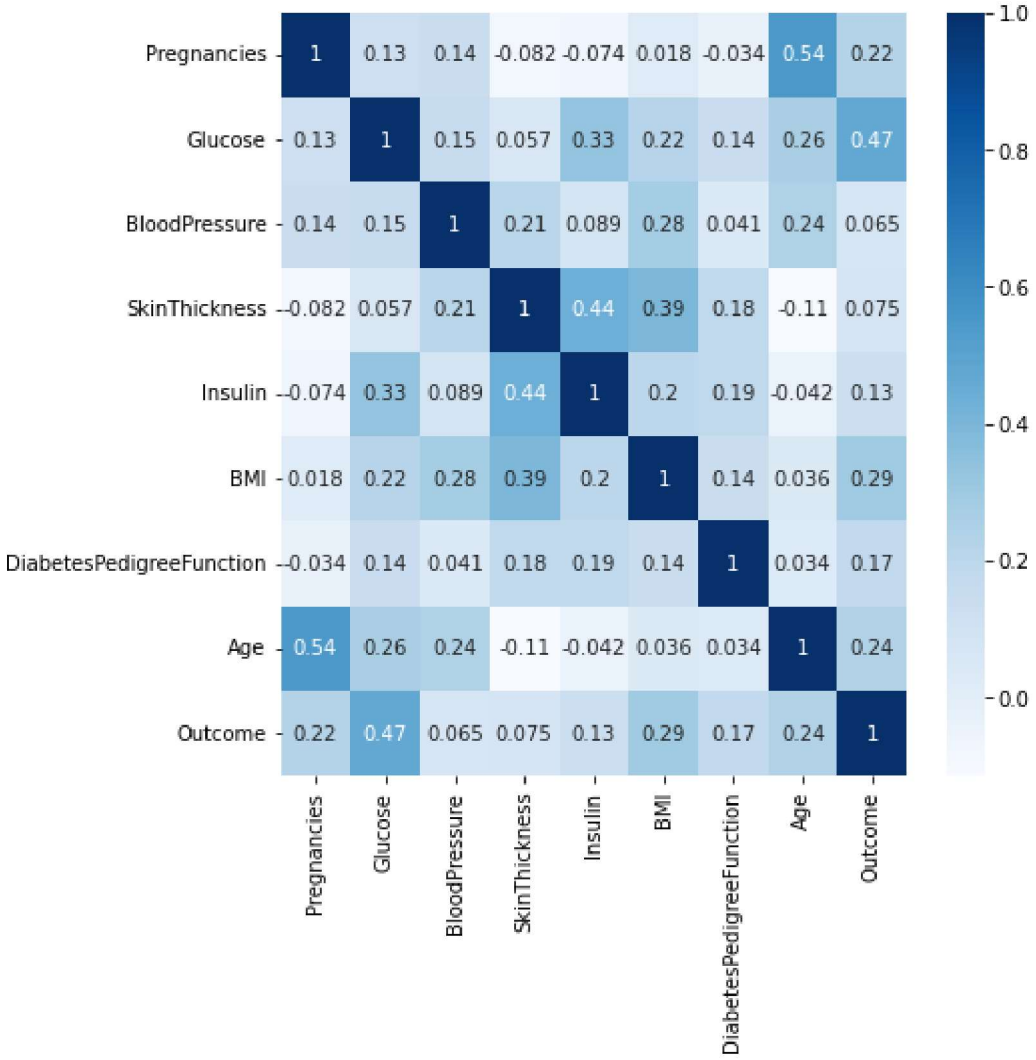
In [11]:
```
plt.figure(figsize=(7,7))
sns.countplot(insulin_bins,hue=data["Outcome"])
```

Out[11]: <AxesSubplot:xlabel='Insulin', ylabel='count'>



In [16]:
```
plt.figure(figsize=(7,7))
sns.heatmap(data.corr(),annot=True,cmap="Blues")
```

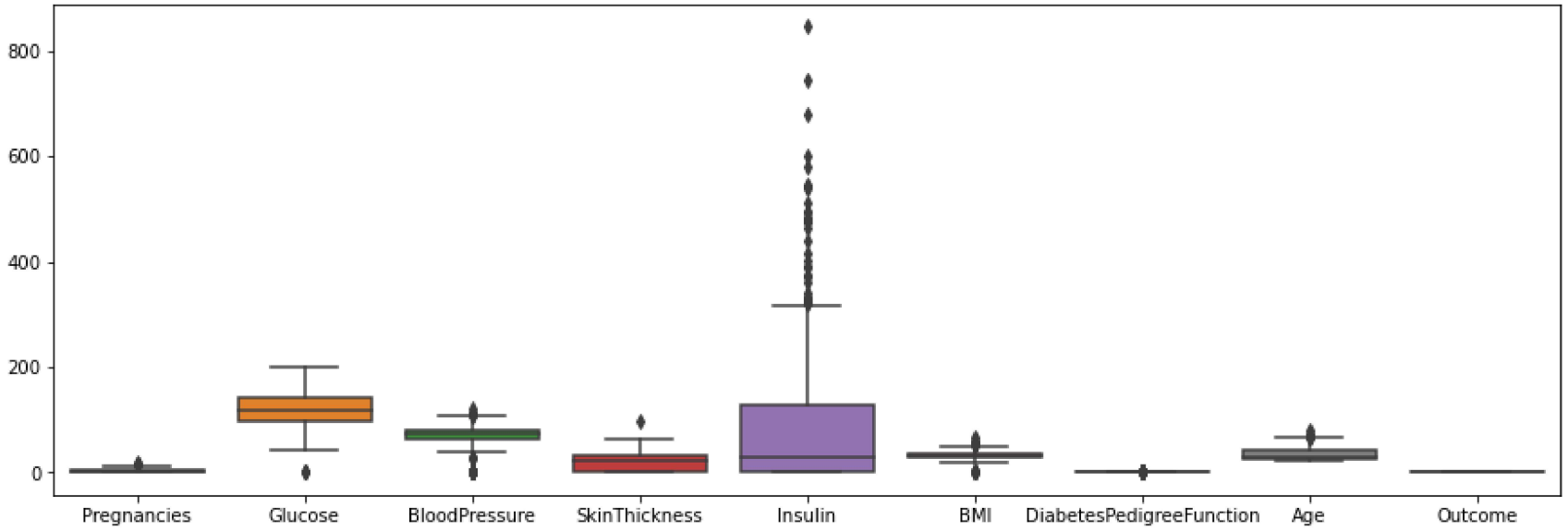
Out[16]: <AxesSubplot:>



Outlier Detection

In [154]:
```
1 plt.figure(figsize=(15,5))
2 sns.boxplot(data=data)
```

Out[154]: <AxesSubplot:>



Train-Test spliting.

In [155]:
```
1 x=data.iloc[:,:-1].values
2 y=data.iloc[:,-1].values
```

In [156]:
```
1 from sklearn.model_selection import train_test_split
2 x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=42,test_size=0.20)
```

In [157]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

Logistic model

In [158]:
```
from sklearn.linear_model import LogisticRegression
classifier=LogisticRegression(random_state=0)
classifier.fit(x_train,y_train)

```

Out[158]:
```
LogisticRegression(random_state=0)
```

In [159]:
```
y_pre=classifier.predict(x_test)
y_pre
```

Out[159]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1,
       0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0],
      dtype=int64)
```

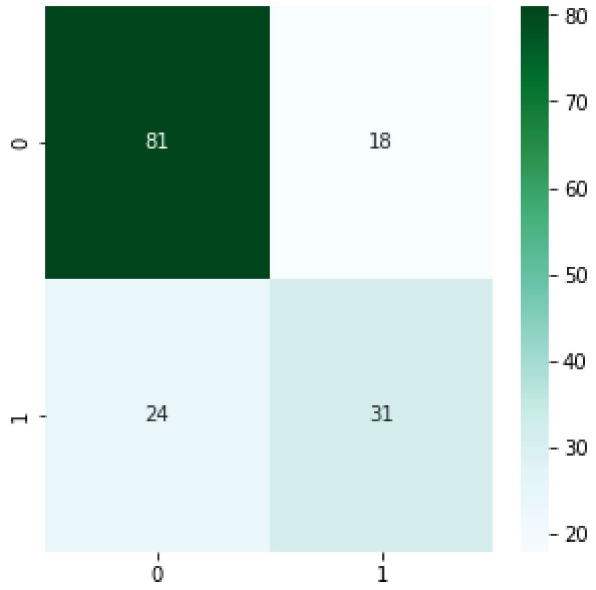
In [160]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
cm=confusion_matrix(y_test,y_pred)
print(cm)
print("accuracy score is")
accuracy_score(y_test,y_pre)*100

```

```
[[81 18]
 [24 31]]
accuracy score is
```

Out[160]:
```
75.32467532467533
```

In [161]:
```
plt.figure(figsize=(5,5))
sns.heatmap(cm,annot=True,cmap="BuGn")
```

Out[161]: <AxesSubplot:>



In [162]:
```
from sklearn.metrics import recall_score,precision_score,roc_auc_score
rs1=recall_score(y_test,y_pre)*100
print("Recall score=",rs)
ps1=precision_score(y_test,y_pre)*100
print("precision score=",ps)
ras1=roc_auc_score(y_test,y_pre)*100
print("ROC score=",ras1)
```

```
Recall score= 56.36363636363636
precision score= 63.26530612244898
ROC score= 73.53535353535354
```

Random Forest

In [163]:
```
from sklearn.ensemble import RandomForestClassifier
rfclassifer=RandomForestClassifier(n_estimators= 10, criterion="entropy")
rfclassifer.fit(x_train,y_train)
```

Out[163]:
```
RandomForestClassifier(criterion='entropy', n_estimators=10)
```

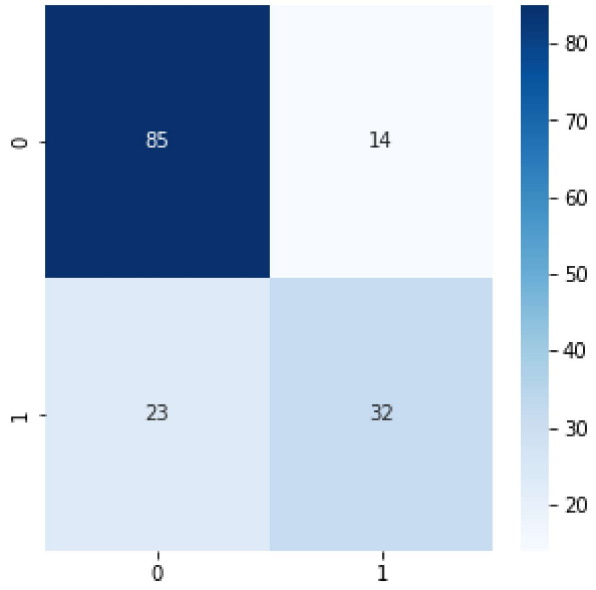
In [164]:
```
y_pred=rfclassifer.predict(x_test)
```

In [165]:
```
from sklearn.metrics import accuracy_score,recall_score,precision_score,confusion_matrix,roc_auc_score
cm=confusion_matrix(y_test,y_pred)
rs=recall_score(y_test,y_pred)*100
ps=precision_score(y_test,y_pred)*100
accs=accuracy_score(y_test,y_pred)*100
ras=roc_auc_score(y_test,y_pred)*100
print("confusion_matrix=\n",cm)
print("recall_score=",rs)
print("precision score=",ps)
print("accuracy_score=",accs)
print("ROC Score=",ras)
```

```
confusion_matrix=
 [[85 14]
 [23 32]]
recall_score= 58.18181818181818
precision score= 69.56521739130434
accuracy_score= 75.97402597402598
ROC Score= 72.020202020202
```

In [166]:
```
plt.figure(figsize=(5,5))
sns.heatmap(cm,cmap="Blues",annot=True)
```

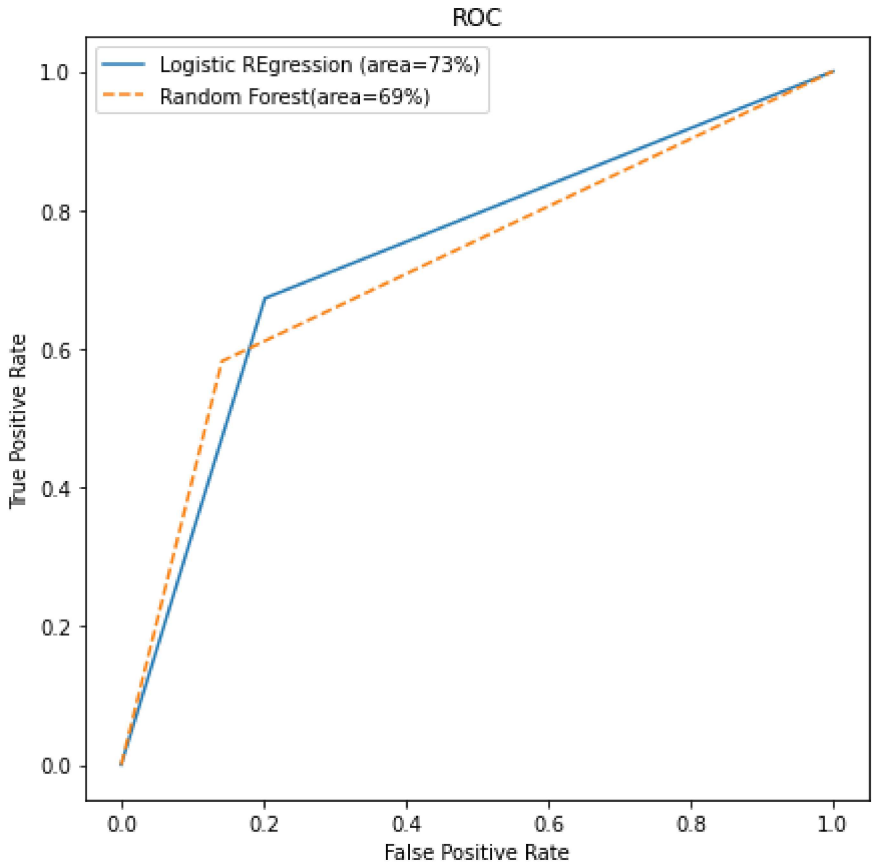
Out[166]: <AxesSubplot:>



ROC Curve

In [167]:
```
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pre)
fprd, tprd, _ = metrics.roc_curve(y_test,  y_pred)
```

In [168]:
```
plt.figure(figsize=(7,7))
plt.plot(fpr,tpr,label='Logistic REgression (area=73%)')
plt.plot(fprd,tprd,"--",label="Random Forest(area=69%)")
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.title("ROC")
plt.legend()
plt.show()
```



In [ ]:
```

```

In [ ]:

```
1
```