diabete_code

October 28, 2021

In [76]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [77]:
```
df=pd.read_csv("diabetes.csv")
df
```

Out[77]:

	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
..	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

[768 rows x 9 columns]

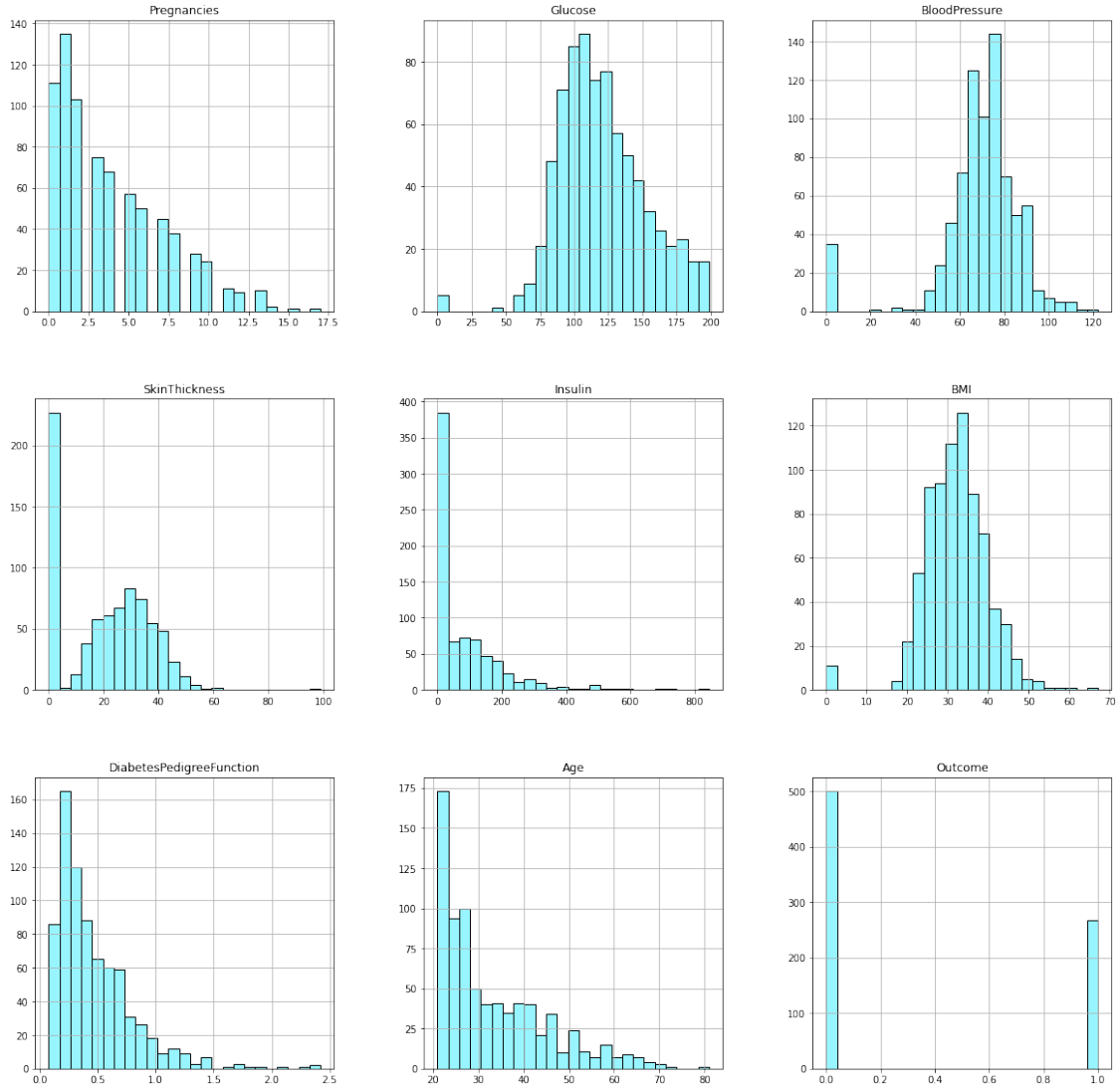
In [78]:
```
df.info()
```

```
<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 [79]: (df== 0).sum()
```

```
Out[79]: Pregnancies                 111
         Glucose                       5
         BloodPressure                35
         SkinThickness               227
         Insulin                     374
         BMI                          11
         DiabetesPedigreeFunction      0
         Age                           0
         Outcome                     500
         dtype: int64
```

```
In [80]: df.hist(figsize = (20,20),bins=25,color='#98F5FF',edgecolor="black")
         plt.show()
```




```
In [81]: df['Glucose'].replace(0,df['Glucose'].mean(),inplace=True)
         df['BloodPressure'].replace(0,df['BloodPressure'].mean(),inplace=True)
         df['SkinThickness'].replace(0,df['SkinThickness'].mean(),inplace=True)
         df['BMI'].replace(0,df['BMI'].mean(),inplace=True)
         df['Insulin'].replace(0,df['Insulin'].median(),inplace=True)

In [82]: df.describe()

Out[82]:        Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
         count   768.000000  768.000000     768.000000     768.000000  768.000000
         mean      3.845052  121.681605      72.254807      26.606479   94.652344
         std       3.369578   30.436016      12.115932       9.631241  105.547598
         min       0.000000   44.000000      24.000000       7.000000   14.000000
         25%       1.000000   99.750000      64.000000      20.536458   30.500000
```

```
50%       3.000000  117.000000      72.000000      23.000000   31.250000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

                BMI  DiabetesPedigreeFunction         Age     Outcome
count   768.000000                768.000000  768.000000  768.000000
mean     32.450805                  0.471876   33.240885    0.348958
std       6.875374                  0.331329   11.760232    0.476951
min      18.200000                  0.078000   21.000000    0.000000
25%      27.500000                  0.243750   24.000000    0.000000
50%      32.000000                  0.372500   29.000000    0.000000
75%      36.600000                  0.626250   41.000000    1.000000
max      67.100000                  2.420000   81.000000    1.000000
```

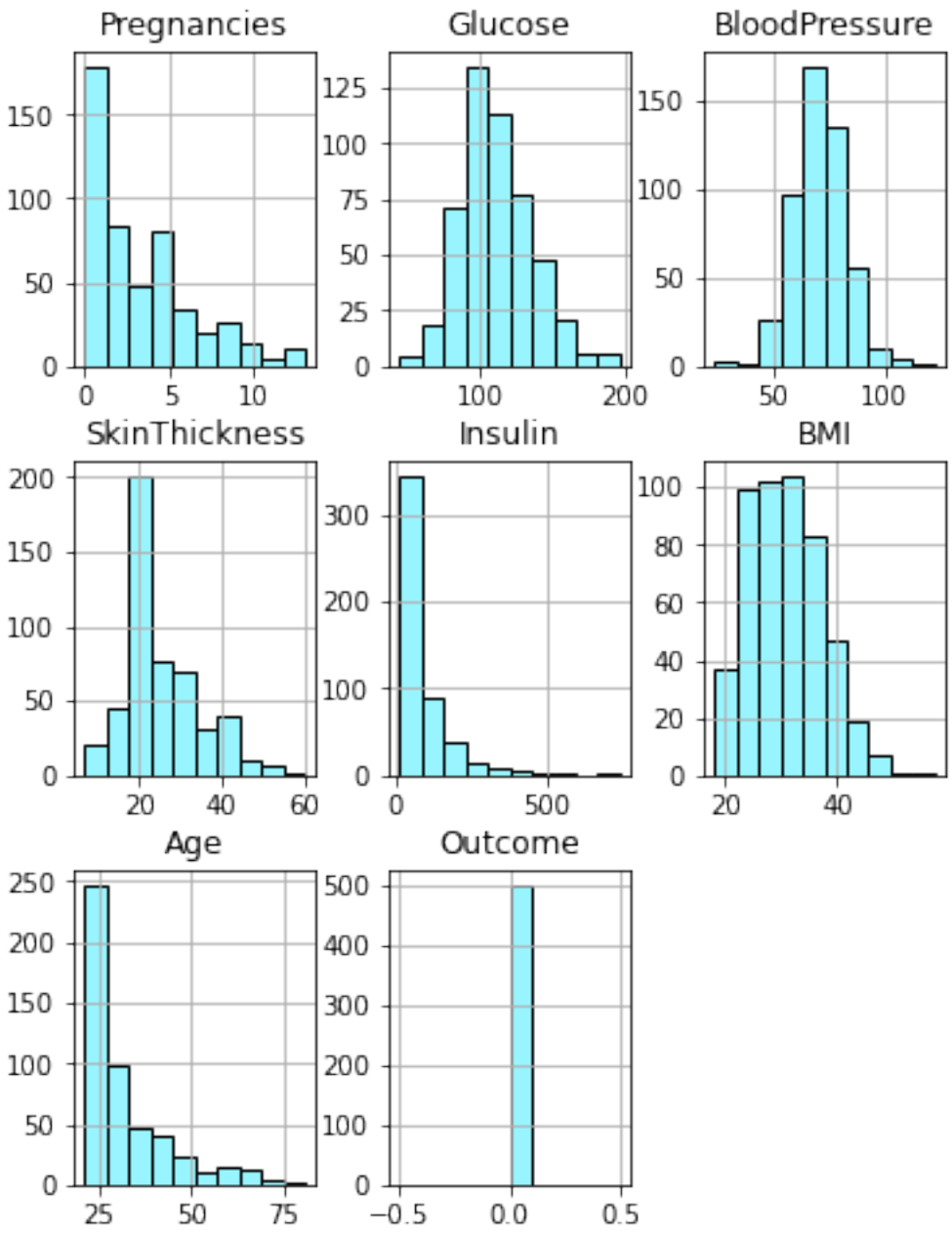
```
In [83]: df.drop(['DiabetesPedigreeFunction'],axis=1,inplace=True)
```

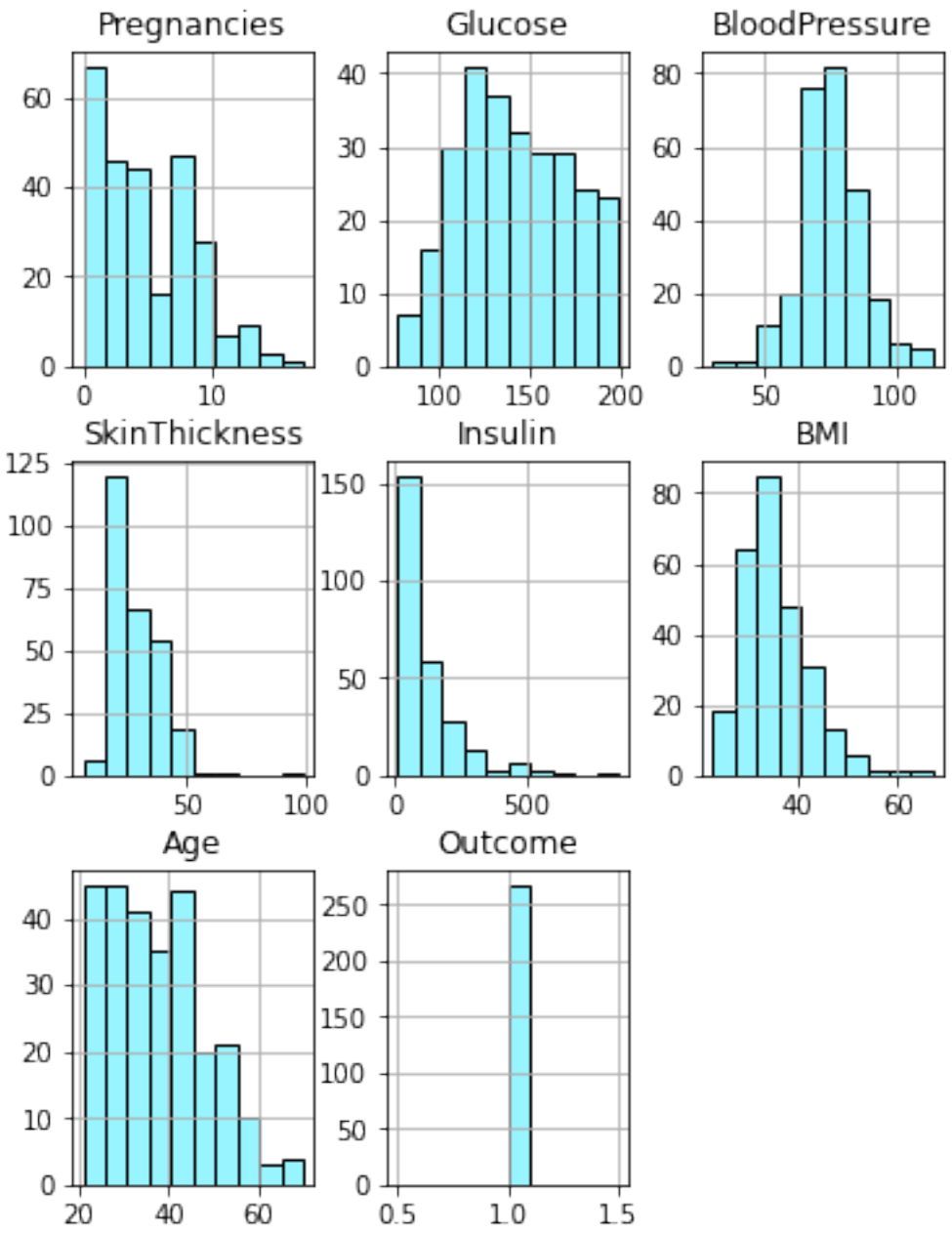
```
In [84]: df.isnull().sum()
```

```
Out[84]: Pregnancies      0
         Glucose          0
         BloodPressure    0
         SkinThickness    0
         Insulin          0
         BMI              0
         Age              0
         Outcome          0
         dtype: int64
```

```
In [85]: df.groupby("Outcome").hist(figsize=(6,8), color="#98F5FF",edgecolor="black")
```

```
Out[85]: Outcome
         0    [[AxesSubplot(0.125,0.670278;0.215278x0.209722...
         1    [[AxesSubplot(0.125,0.670278;0.215278x0.209722...
         dtype: object
```


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
Age
Outcome




```
In [86]: df.corr()
```

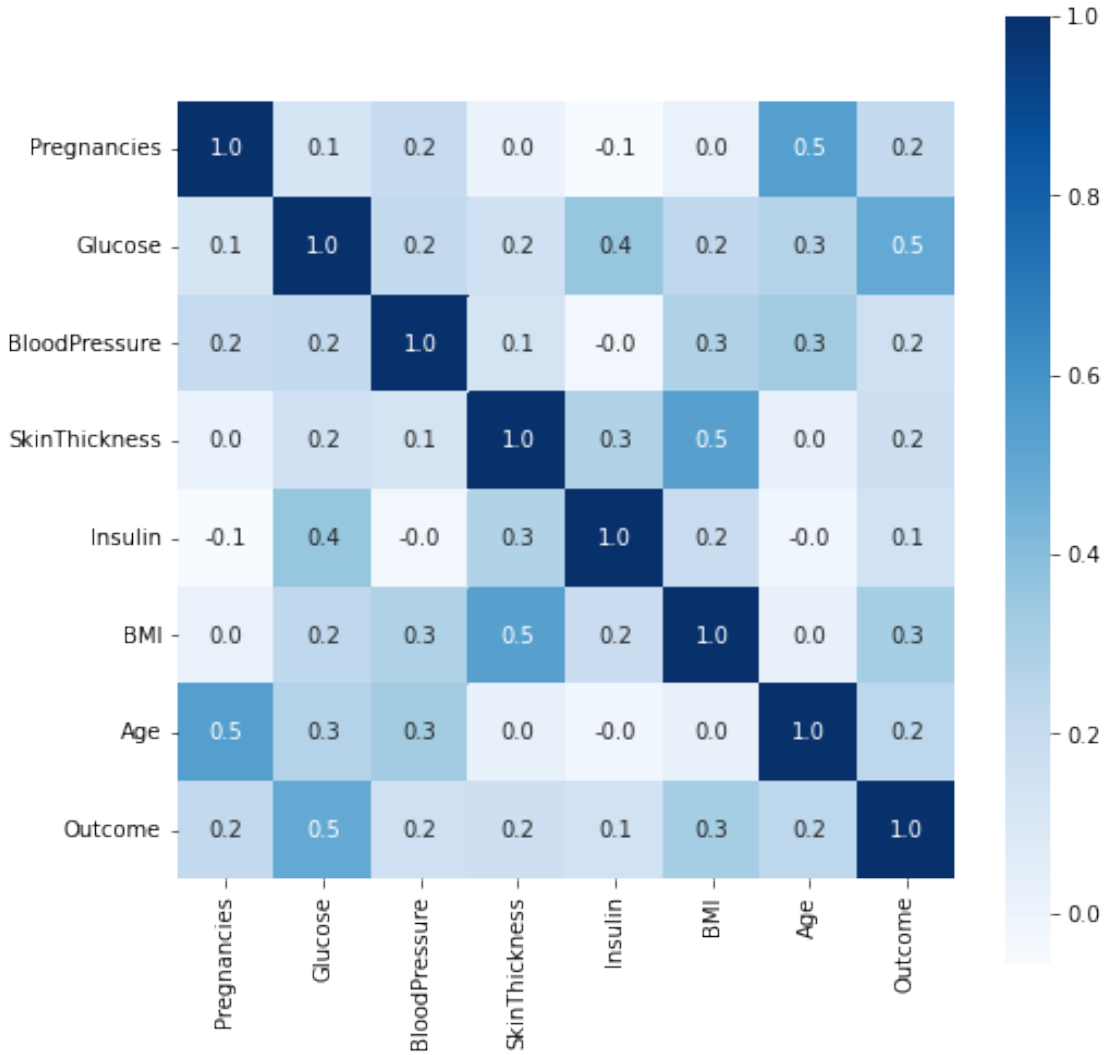
Out[86]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	\
Pregnancies	1.000000	0.127964	0.208984	0.013376	-0.055697	
Glucose	0.127964	1.000000	0.219666	0.160766	0.357081	
BloodPressure	0.208984	0.219666	1.000000	0.134155	-0.022049	
SkinThickness	0.013376	0.160766	0.134155	1.000000	0.274253	
Insulin	-0.055697	0.357081	-0.022049	0.274253	1.000000	
BMI	0.021546	0.231478	0.281231	0.535703	0.189031	

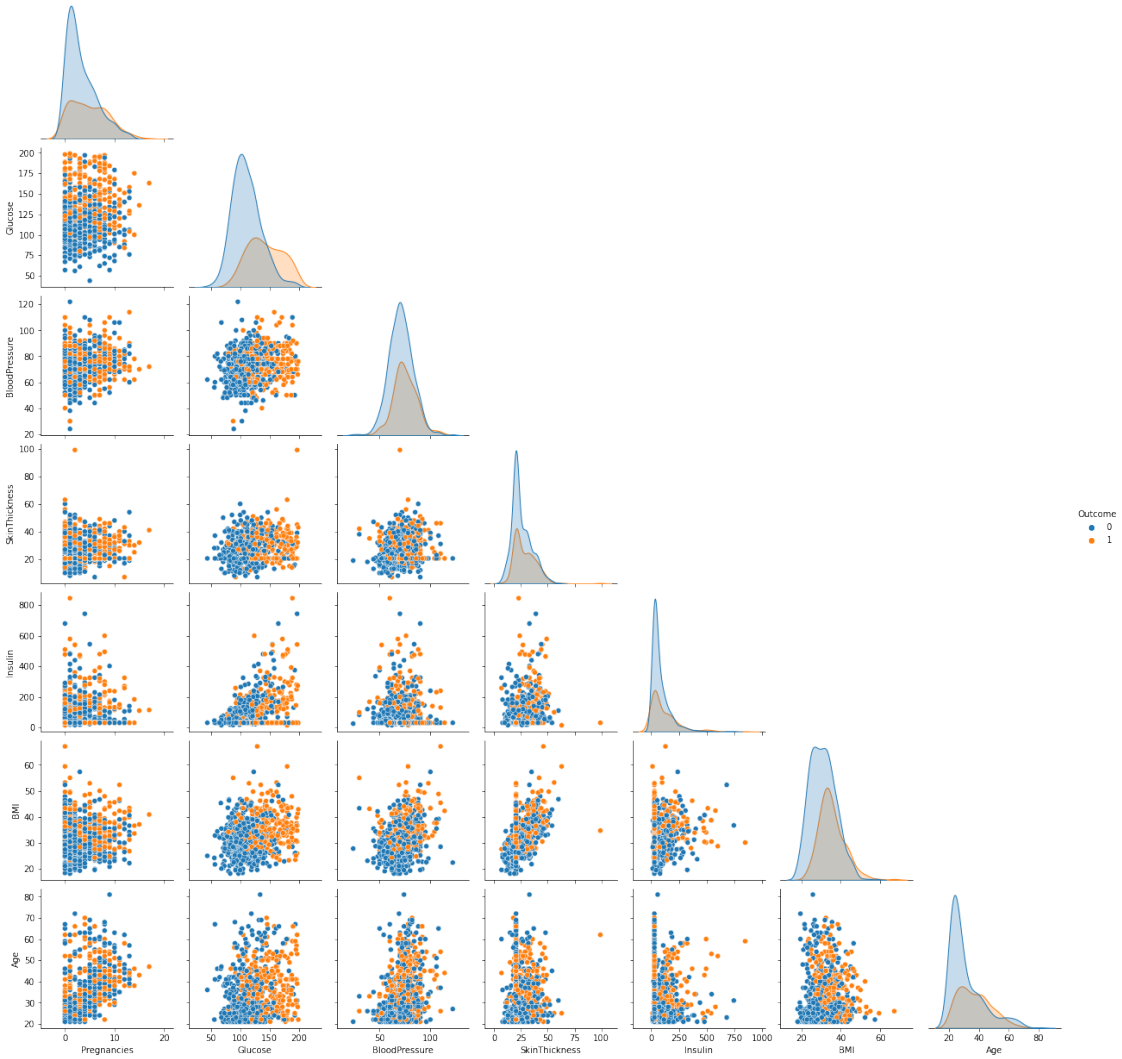
```
Age                       0.544341  0.266600       0.326740       0.026423 -0.015413
Outcome                   0.221898  0.492908       0.162986       0.175026  0.148457

                              BMI       Age   Outcome
Pregnancies              0.021546  0.544341  0.221898
Glucose                  0.231478  0.266600  0.492908
BloodPressure            0.281231  0.326740  0.162986
SkinThickness            0.535703  0.026423  0.175026
Insulin                  0.189031 -0.015413  0.148457
BMI                      1.000000  0.025748  0.312254
Age                      0.025748  1.000000  0.238356
Outcome                  0.312254  0.238356  1.000000
```

In [88]:
```
corr = df.corr()
plt.figure(figsize=(8,8))
sns.heatmap(corr,cbar=True,square=True,fmt='.1f',annot=True,cmap='Blues')
```

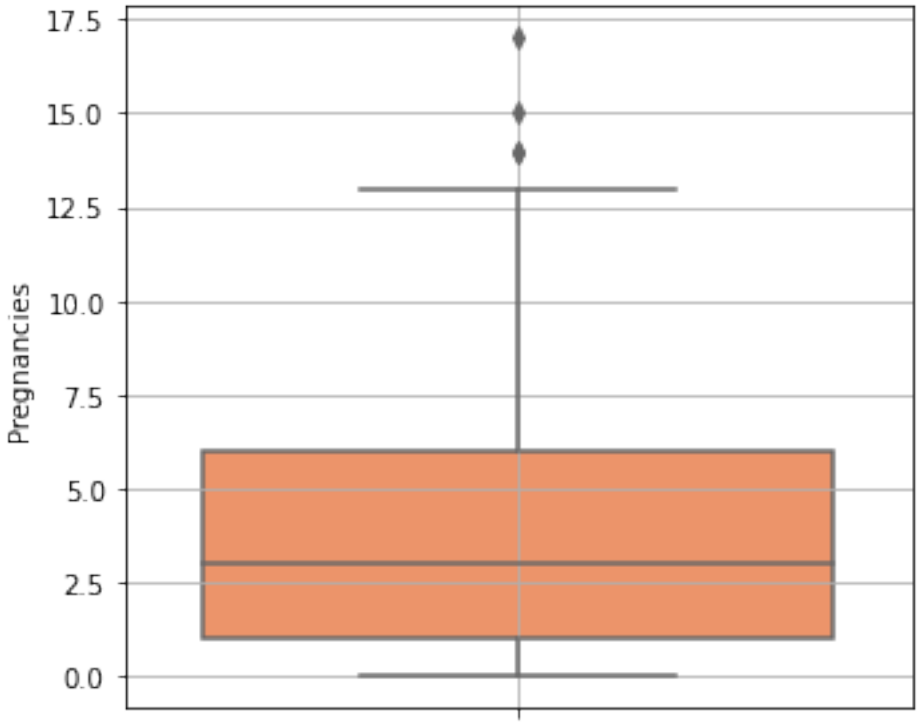
Out[88]: <AxesSubplot:>



```
In [89]: sns.pairplot(df,hue='Outcome',corner=True)
         sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
         plt.show()
```




```
In [90]: plt.figure(figsize=(5,4))
         sns.boxplot(y=df['Pregnancies'],palette='OrRd_r',saturation=0.8)
         plt.tight_layout()
         plt.grid(True)
         plt.show()
```

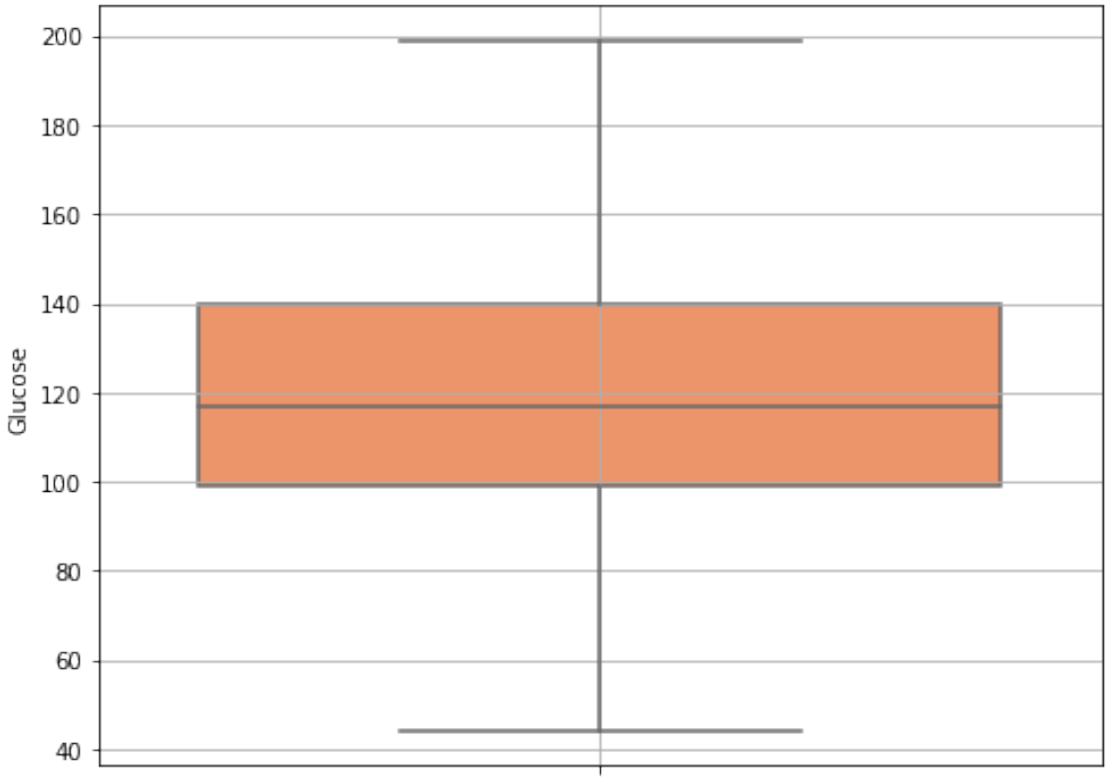



```
In [91]: df[df['Pregnancies']>13]
```

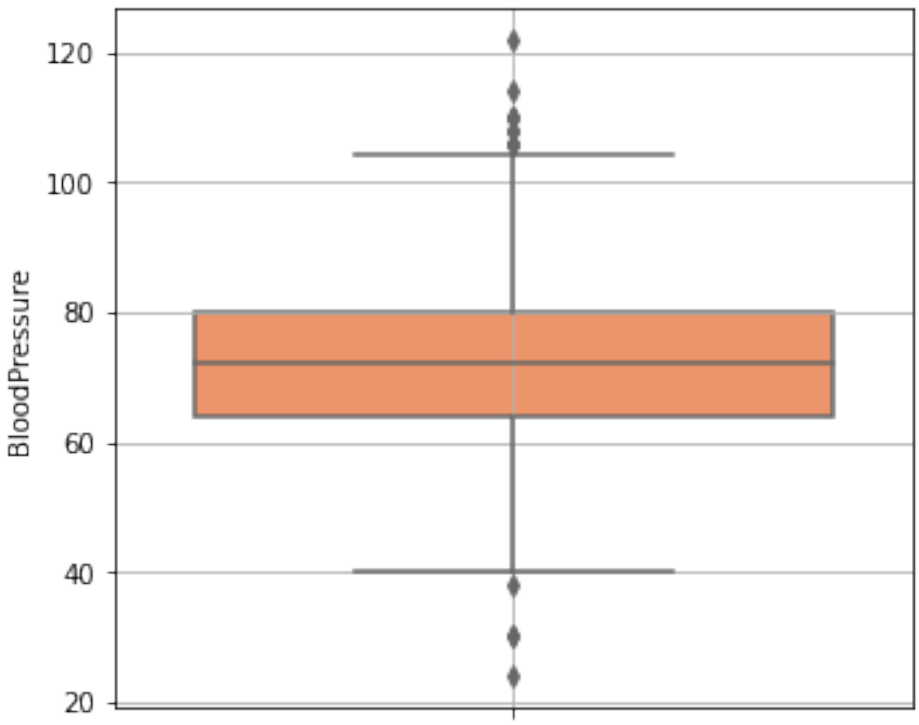
Out[91]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Age	Outcome
88	15	136.0	70.0	32.0	110.0	37.1	43	1
159	17	163.0	72.0	41.0	114.0	40.9	47	1
298	14	100.0	78.0	25.0	184.0	36.6	46	1
455	14	175.0	62.0	30.0	30.5	33.6	38	1

```
In [92]: df.drop(df[df['Pregnancies']>13].index, inplace=True)

In [93]: plt.figure(figsize=(7,5))
         sns.boxplot(y=df['Glucose'],palette='OrRd_r',saturation=0.8)
         plt.tight_layout()
         plt.grid(True)
         plt.show()
```




```
In [94]: plt.figure(figsize=(5,4))
         sns.boxplot(y=df['BloodPressure'],palette='OrRd_r',saturation=0.8)
         plt.tight_layout()
         plt.grid(True)
         plt.show()
```

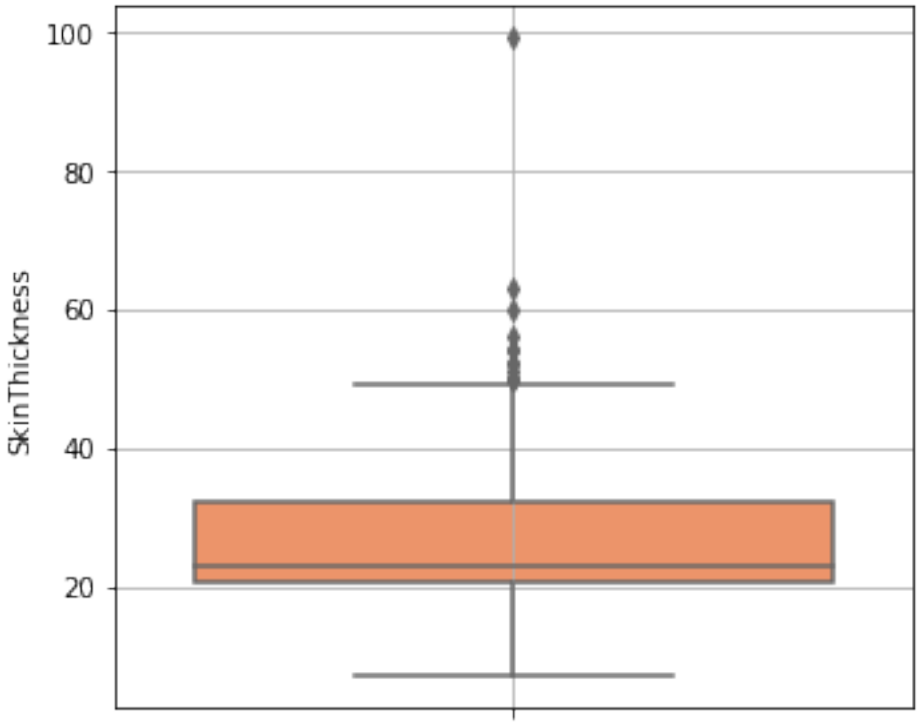



```
In [95]: df[(df['BloodPressure']<40)]
```

Out[95]:

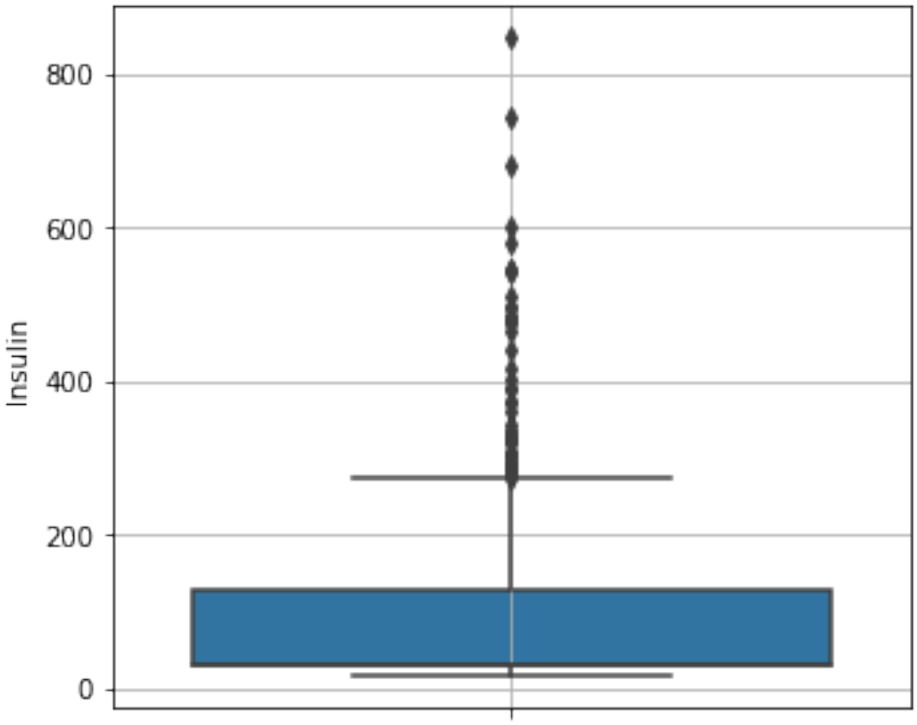
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Age	Outcome
18	1	103.0	30.0	38.0	83.0	43.3	33	0
125	1	88.0	30.0	42.0	99.0	55.0	26	1
597	1	89.0	24.0	19.0	25.0	27.8	21	0
599	1	109.0	38.0	18.0	120.0	23.1	26	0

```
In [96]: df['BloodPressure'].replace(30.0,40,inplace=True)
         df['BloodPressure'].replace(38.0,40,inplace=True)
         df['BloodPressure'].replace(24.0,40,inplace=True)

In [97]: plt.figure(figsize=(5,4))
         sns.boxplot(y=df['SkinThickness'],palette='OrRd_r',saturation=0.8)
         plt.tight_layout()
         plt.grid(True)
         plt.show()
```

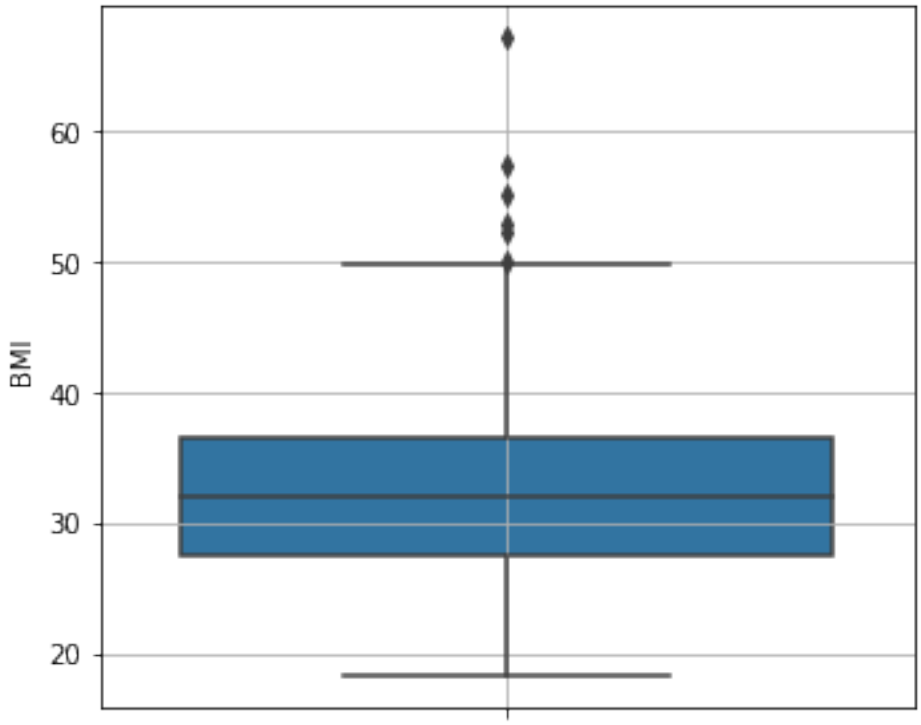



```
In [98]: df.drop(df[df['SkinThickness']>55].index, inplace=True)
```

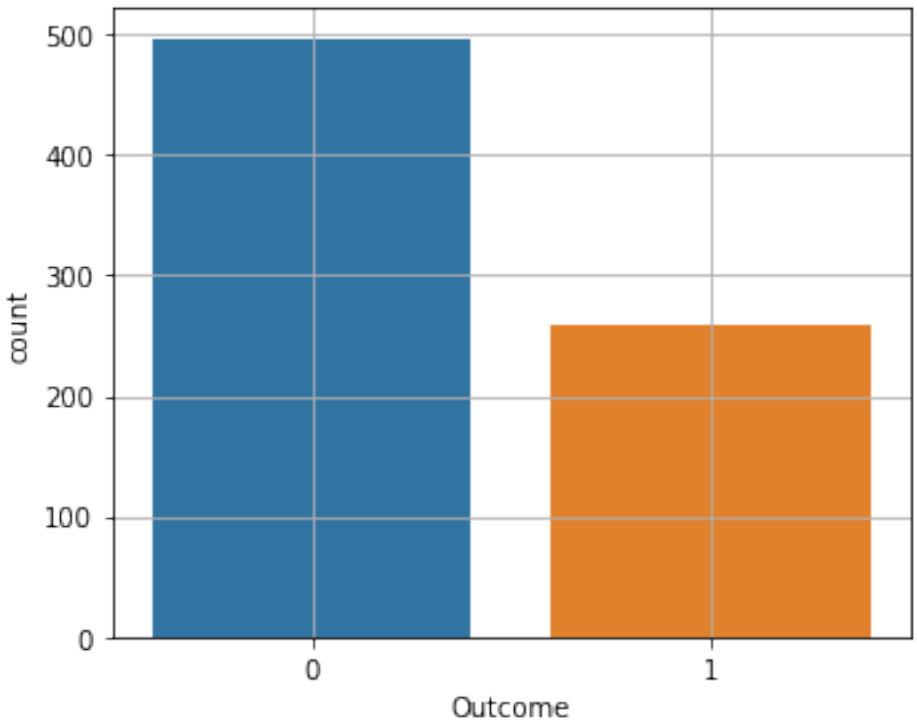
```
In [99]: plt.figure(figsize=(5,4))
         sns.boxplot(y=df['Insulin'])
         sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
         plt.tight_layout()
         plt.grid(True)
         plt.show()
```



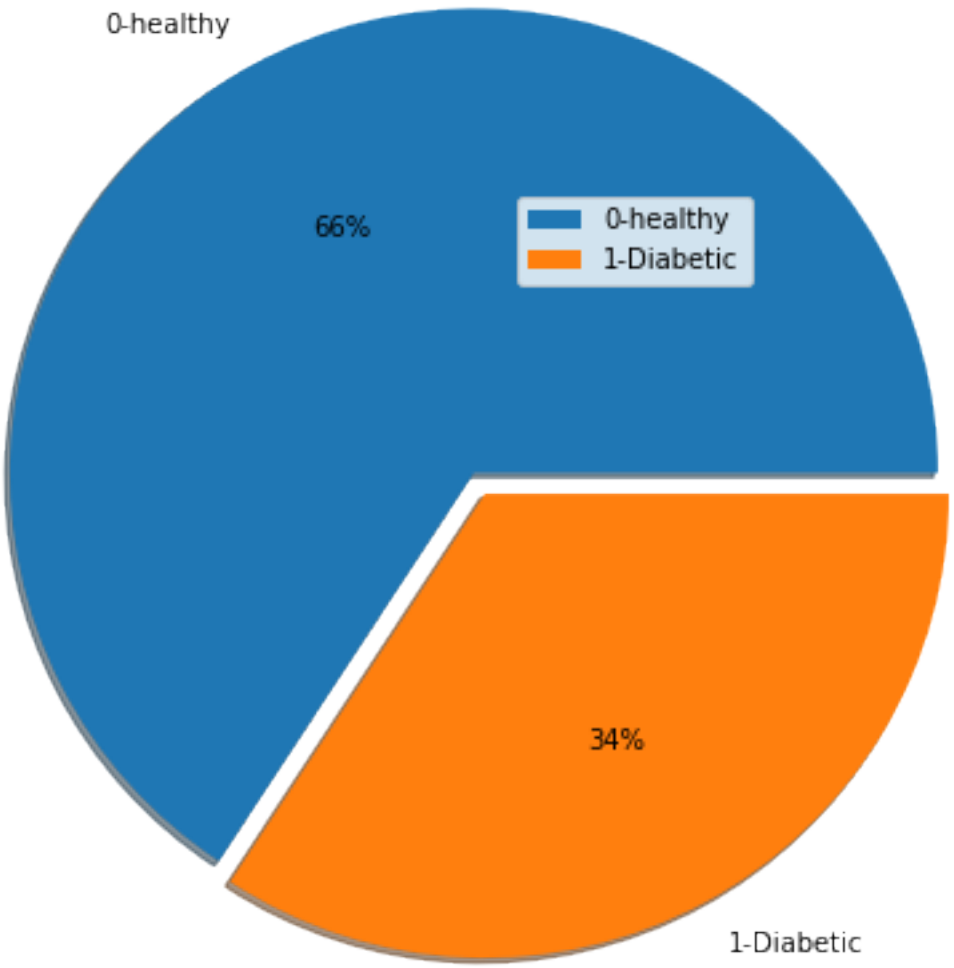

```
In [100]: df.drop(df[df['Insulin']>600].index, inplace=True)

In [101]: plt.figure(figsize=(5,4))
          sns.boxplot(y=df['BMI'])
          sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
          plt.tight_layout()
          plt.grid(True)
          plt.show()
```



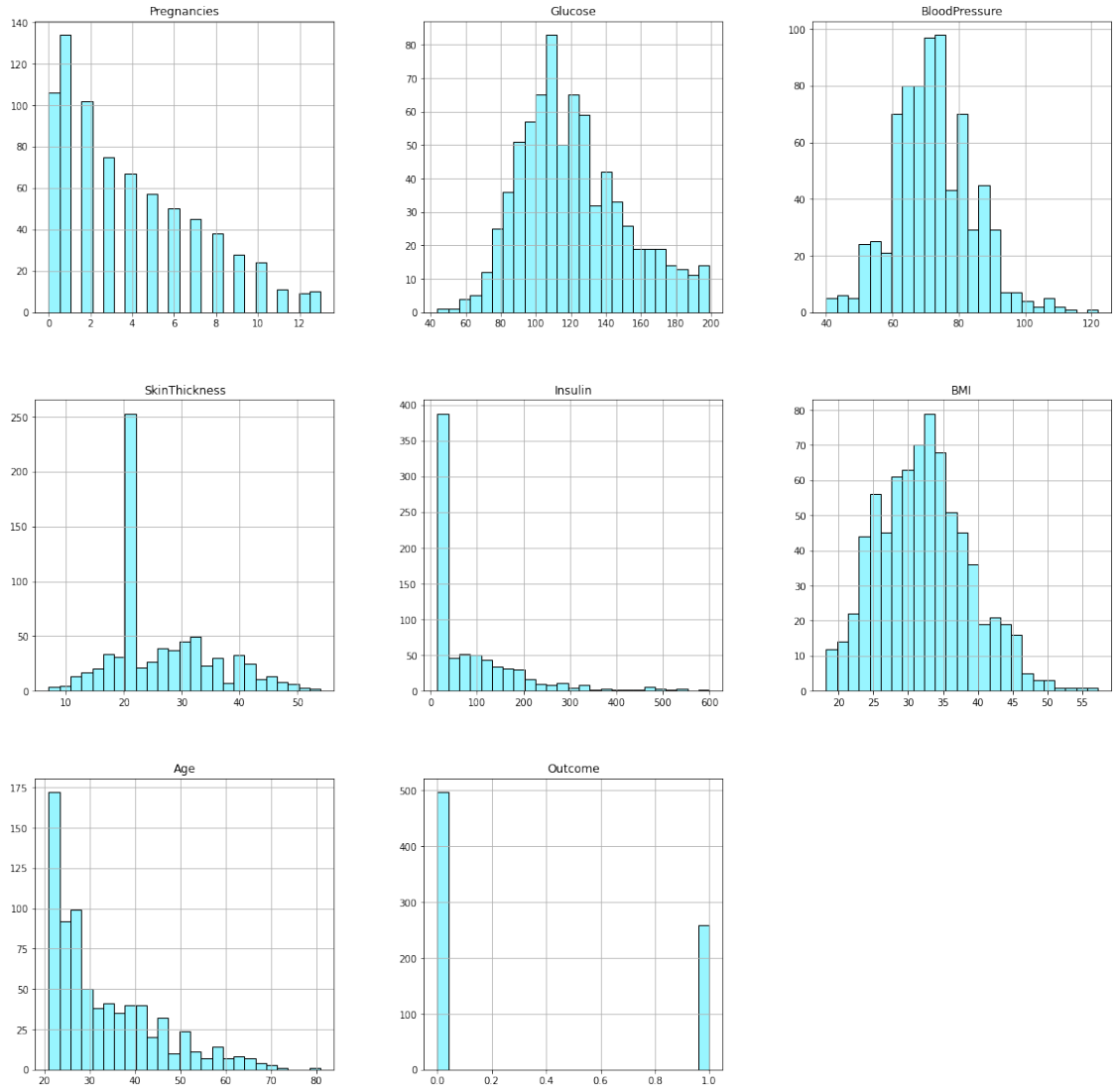

```
In [102]: df.drop(df[df['BMI']>60].index, inplace=True)

In [103]: df['Outcome'].value_counts()

Out[103]: 0    497
          1    259
          Name: Outcome, dtype: int64

In [104]: plt.figure(figsize=(5,4))
          sns.countplot(df['Outcome'])
          sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
          plt.tight_layout()
          plt.grid(True)
          plt.show()

/usr/local/lib/python3.8/site-packages/seaborn/_decorators.py:36: FutureWarning: Pass the follo
  warnings.warn(
```



In [105]:
```
out_df=pd.DataFrame(df.groupby('Outcome')['Outcome'].count())

plt.pie(out_df['Outcome'],labels=['0-healthy','1-Diabetic'],autopct='%.0f%%',radius=
plt.legend()
plt.show()
```

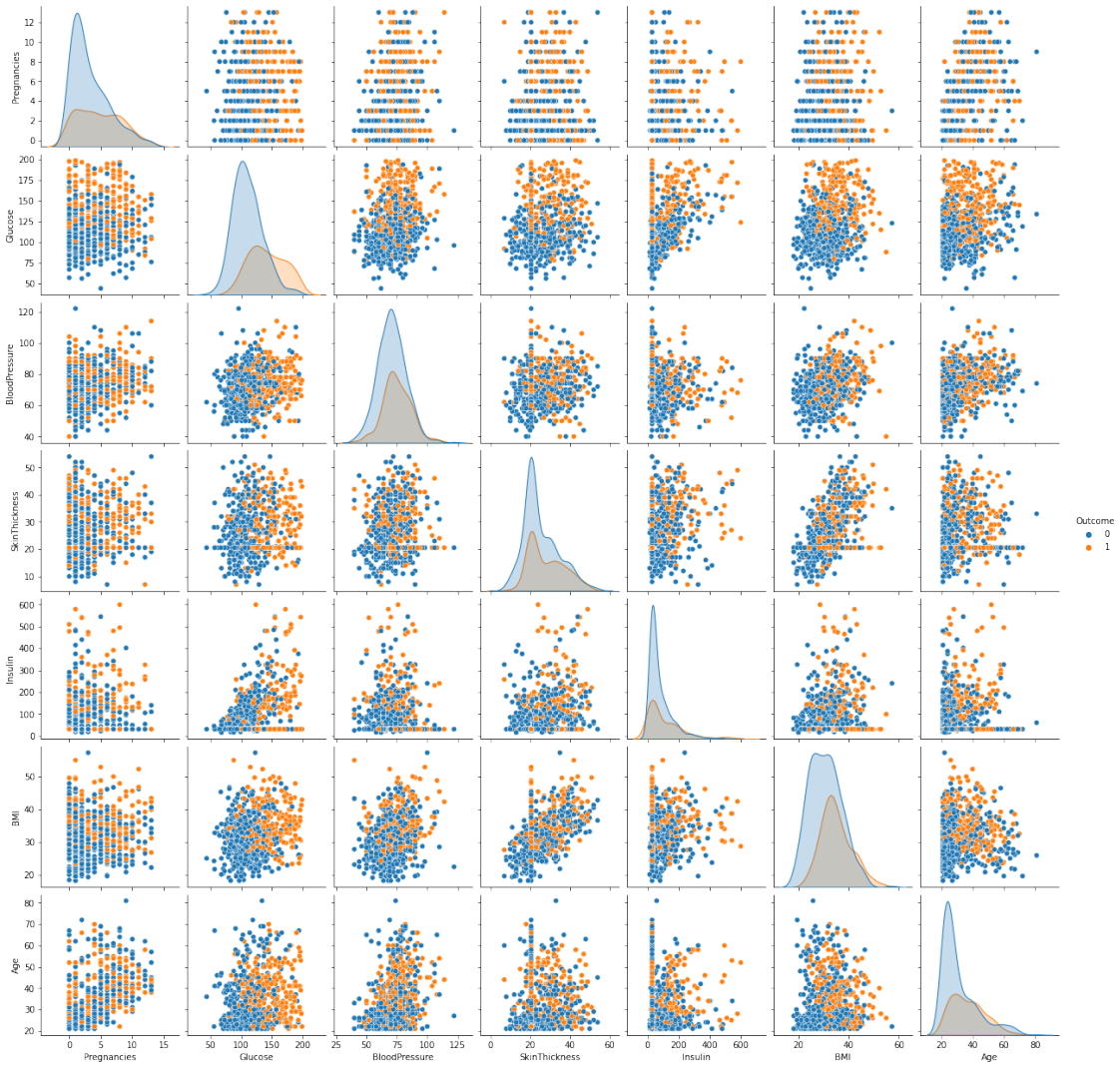



```
In [106]: df.hist(figsize = (20,20),bins=25,color='#98F5FF',edgecolor="black")
          plt.show()
```

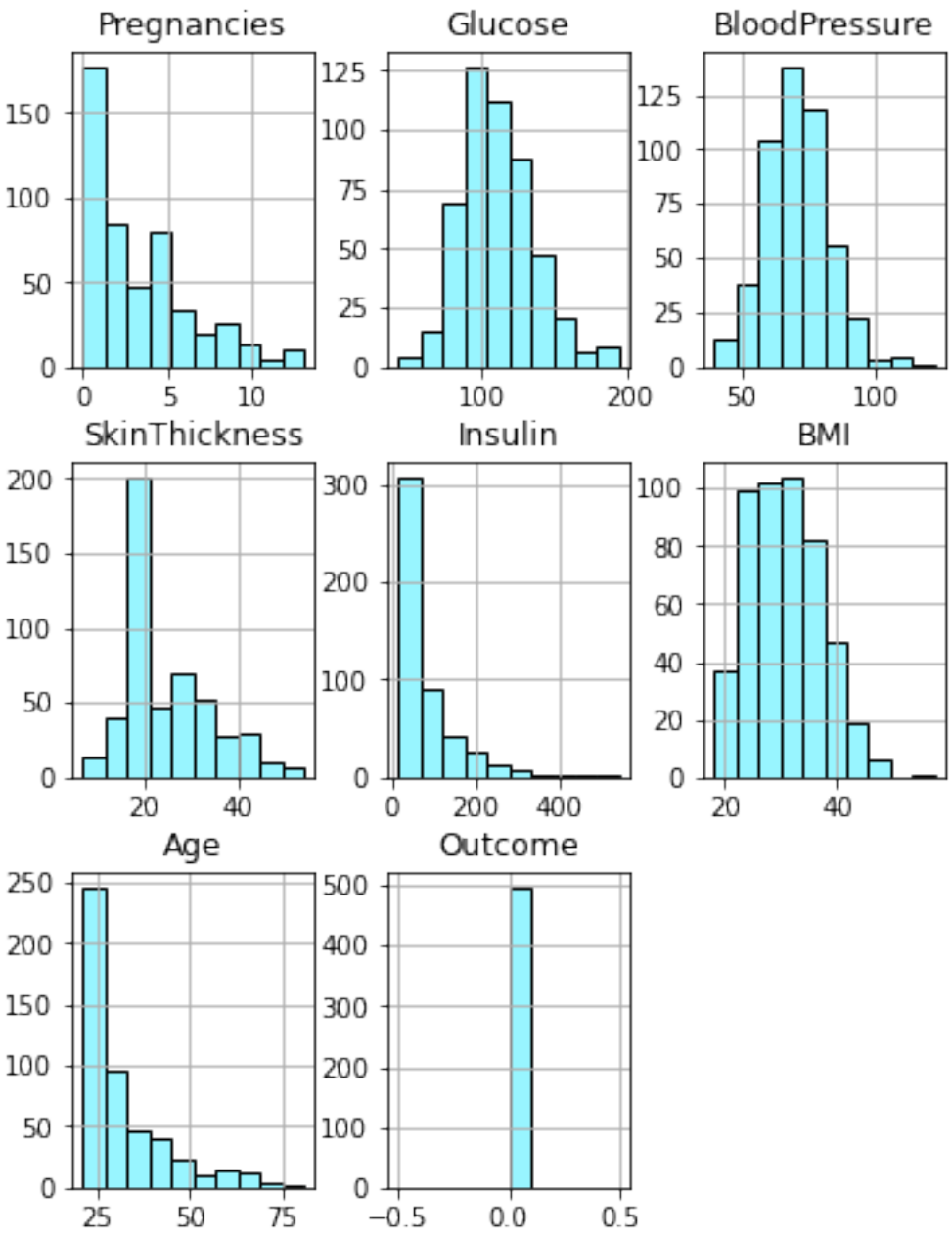


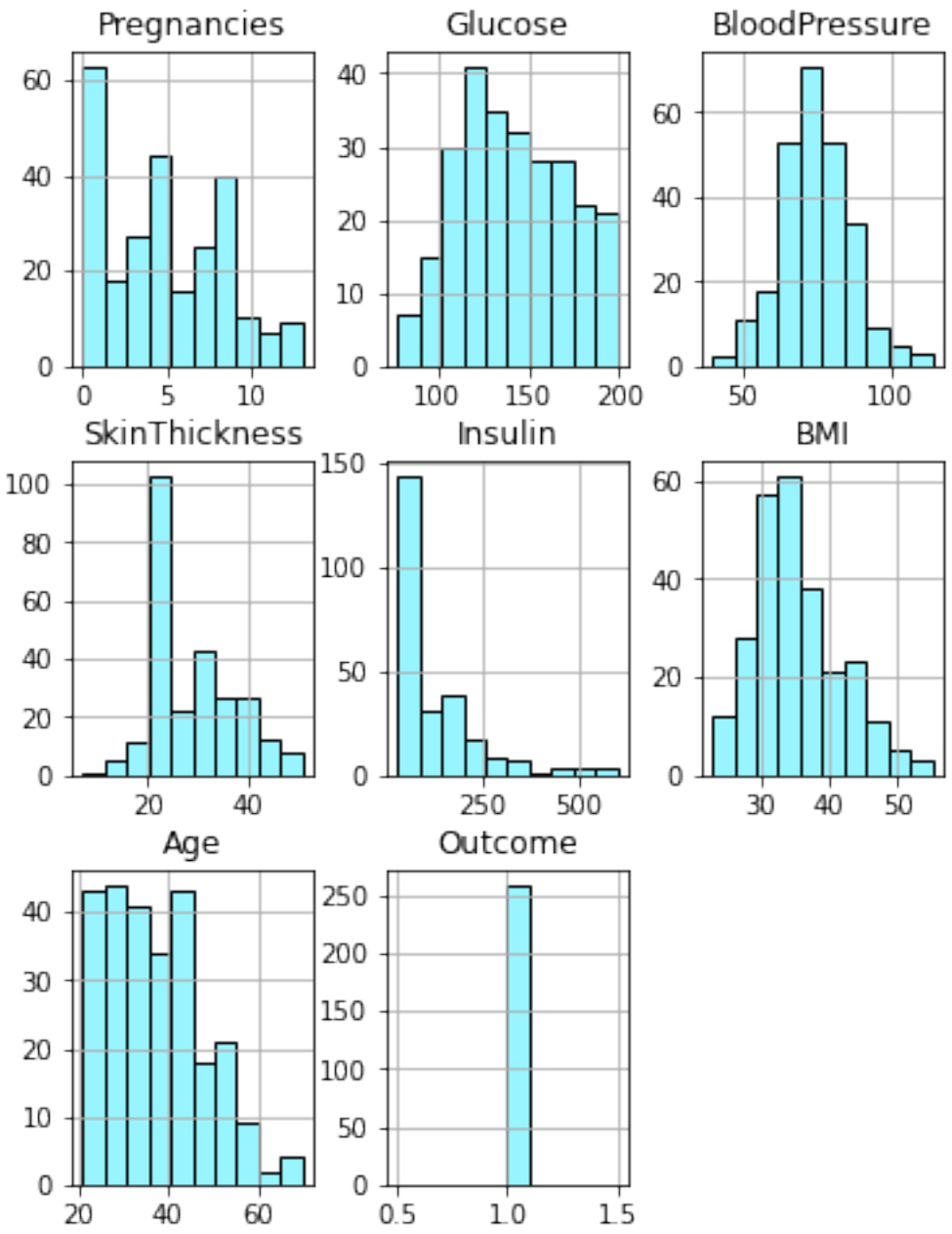

```
In [108]: sns.pairplot(df,hue='Outcome')
```

```
Out[108]: <seaborn.axisgrid.PairGrid at 0x7f714a589a00>
```



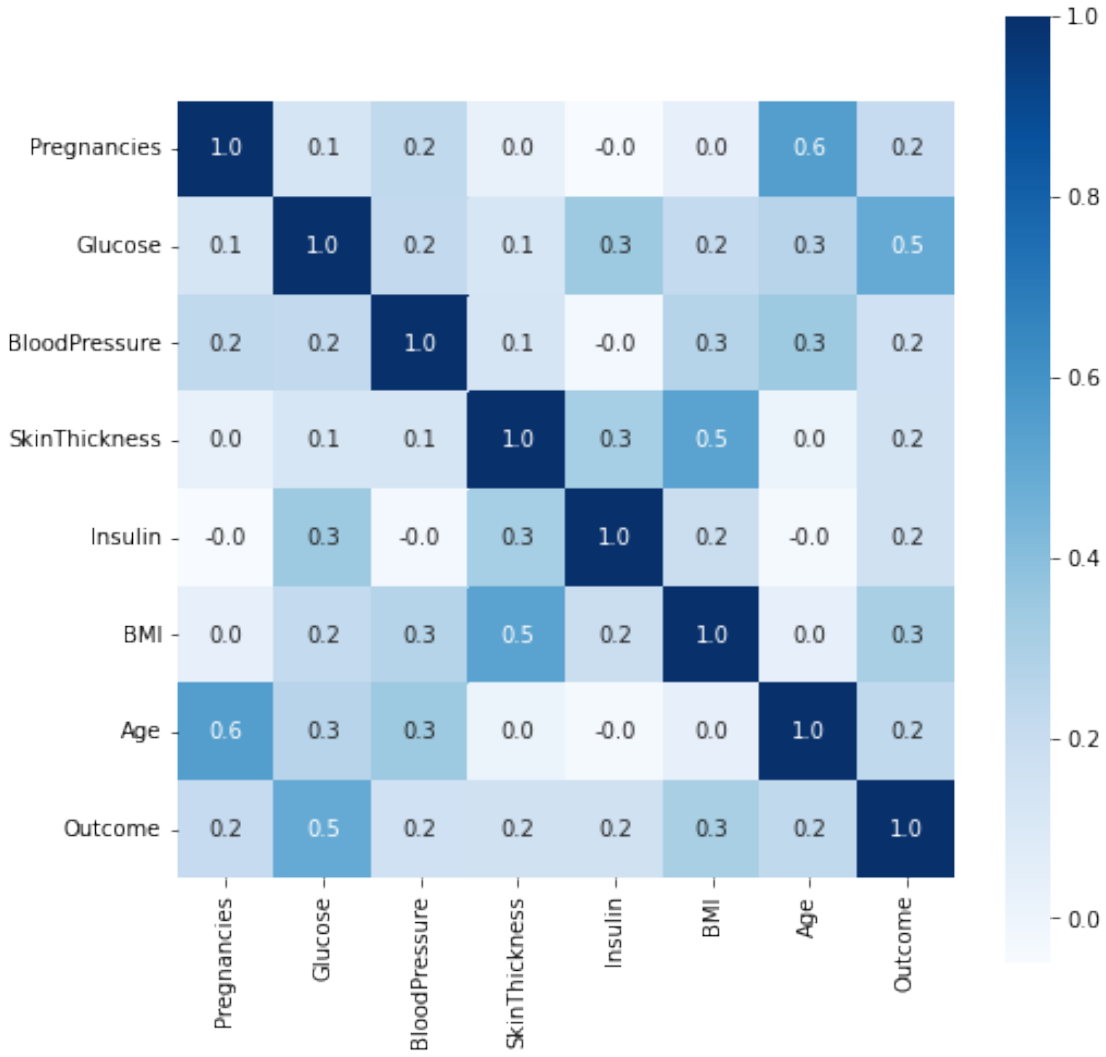

```
In [109]: df.groupby("Outcome").hist(figsize=(6,8), color="#98F5FF",edgecolor="black")

Out[109]: Outcome
          0    [[AxesSubplot(0.125,0.670278;0.215278x0.209722...
          1    [[AxesSubplot(0.125,0.670278;0.215278x0.209722...
          dtype: object
```


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
Age
Outcome




```
In [110]: corr = df.corr()
          plt.figure(figsize=(8,8))
          sns.heatmap(corr,cbar=True,square=True,fmt='.1f',annot=True,cmap='Blues')
```

```
Out[110]: <AxesSubplot:>
```

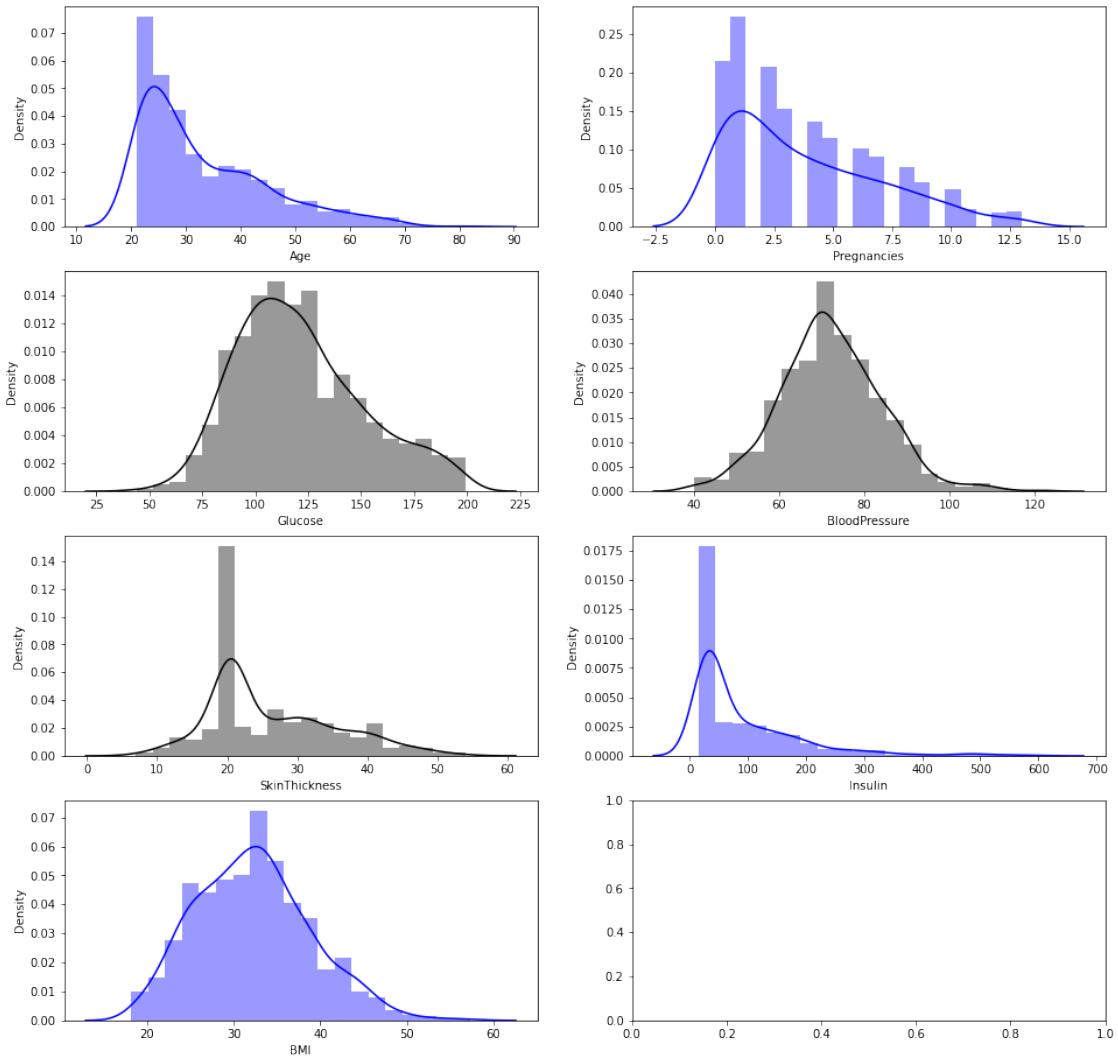



```
In [111]: fig, ax = plt.subplots(4,2, figsize=(16,16))
          sns.distplot(df['Age'], bins = 20, ax=ax[0,0] , color="blue")
          sns.distplot(df['Pregnancies'], bins = 20, ax=ax[0,1], color="blue")
          sns.distplot(df['Glucose'], bins = 20, ax=ax[1,0], color="black")
          sns.distplot(df['BloodPressure'], bins = 20, ax=ax[1,1], color="black")
          sns.distplot(df['SkinThickness'], bins = 20, ax=ax[2,0], color="black")
          sns.distplot(df['Insulin'], bins = 20, ax=ax[2,1], color="blue")

          sns.distplot(df['BMI'], bins = 20, ax=ax[3,0],color="blue")
```

```
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
```

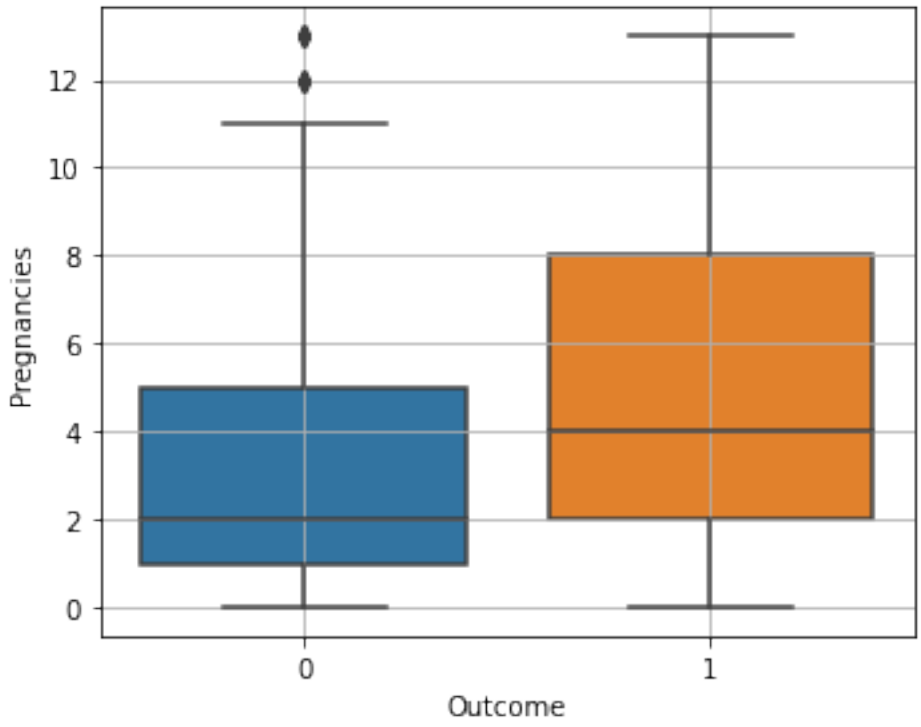
```
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
/usr/local/lib/python3.8/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot
  warnings.warn(msg, FutureWarning)
```

Out[111]: <AxesSubplot:xlabel='BMI', ylabel='Density'>



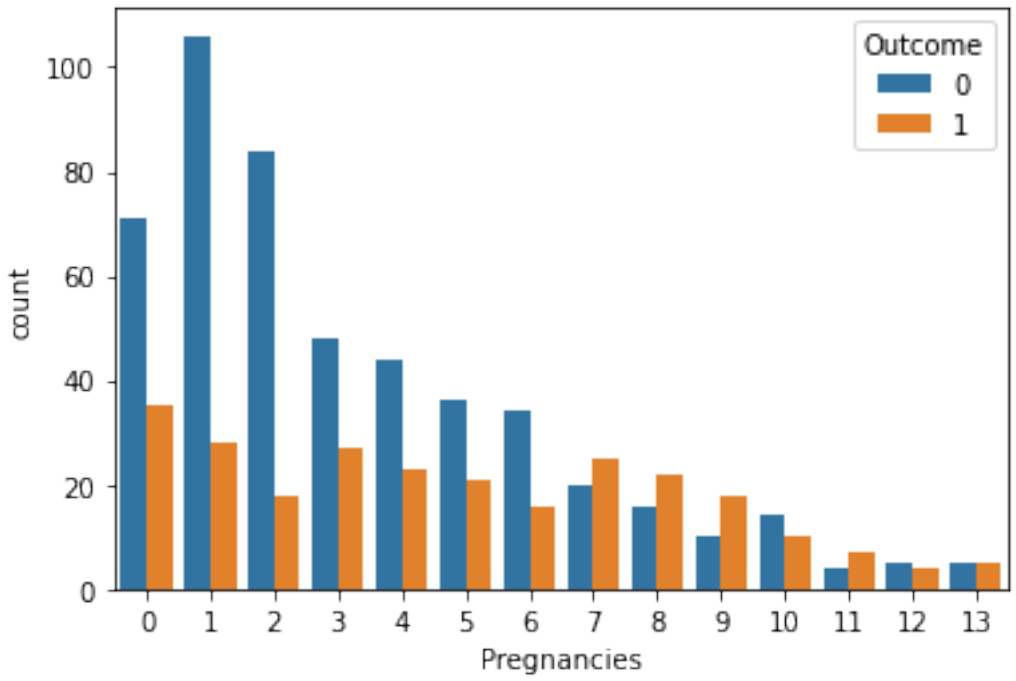
In [112]:
```
plt.figure(figsize=(5,4))
sns.boxplot(y=df['Pregnancies'],x=df['Outcome'])
```

```
sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
plt.tight_layout()
plt.grid(True)
plt.show()
```

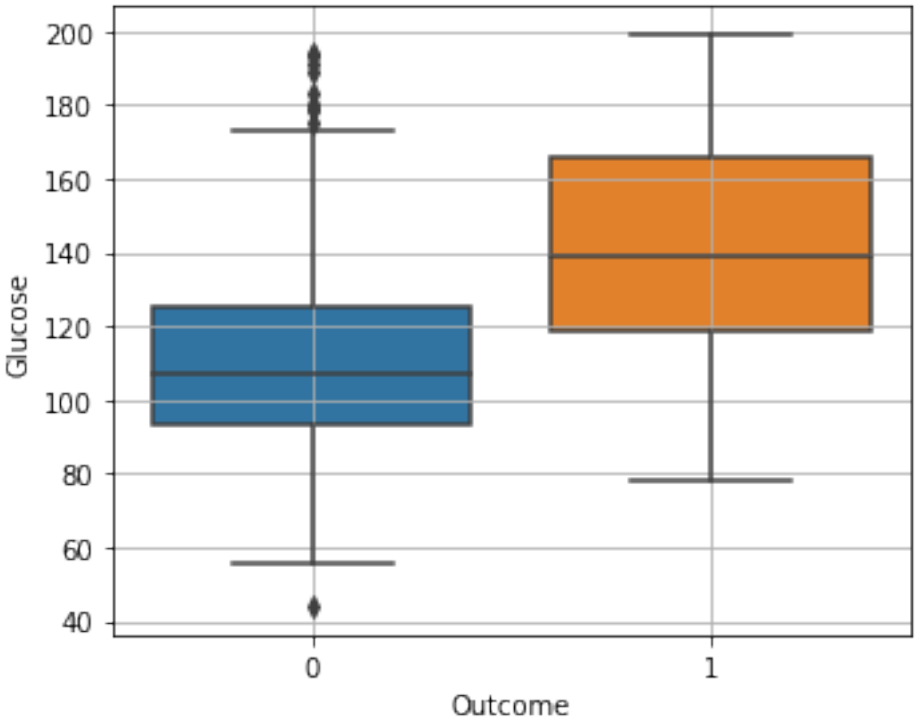



```
In [113]: sns.countplot(x=df['Pregnancies'],hue=df['Outcome'])
```

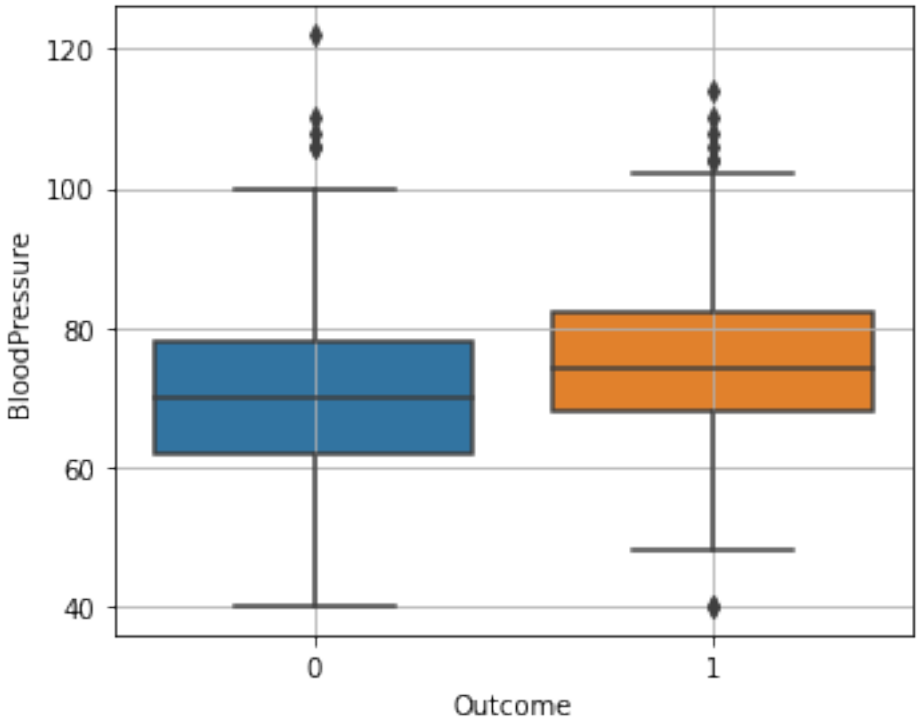
```
Out[113]: <AxesSubplot:xlabel='Pregnancies', ylabel='count'>
```



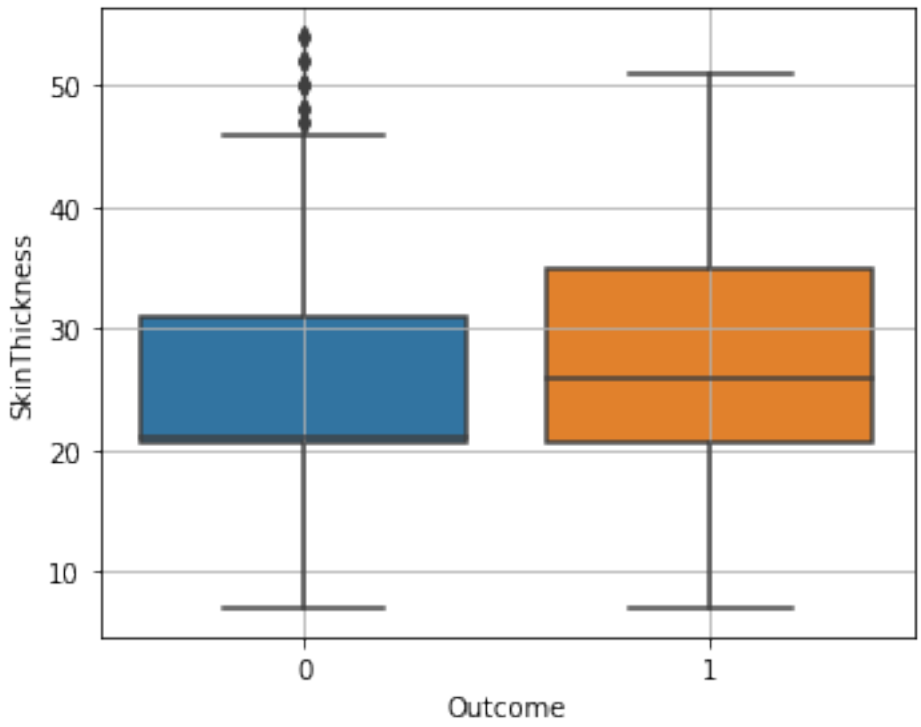

```
In [114]: plt.figure(figsize=(5,4))
          sns.boxplot(y=df['Glucose'],x=df['Outcome'])
          sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
          plt.tight_layout()
          plt.grid(True)
          plt.show()
```



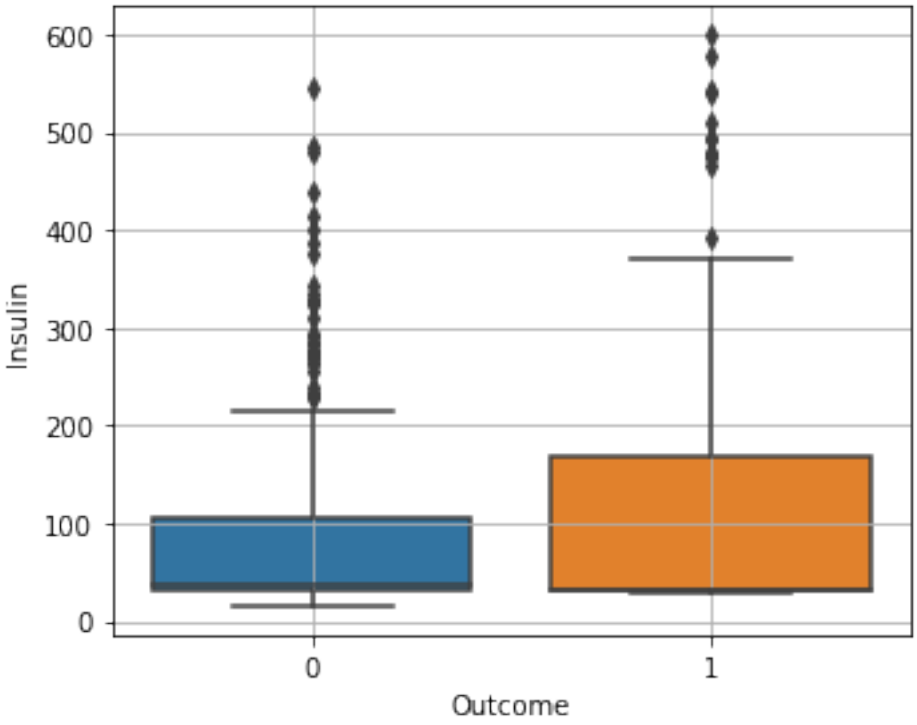

```
In [115]: plt.figure(figsize=(5,4))
          sns.boxplot(y=df['BloodPressure'],x=df['Outcome'])
          sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
          plt.tight_layout()
          plt.grid(True)
          plt.show()
```



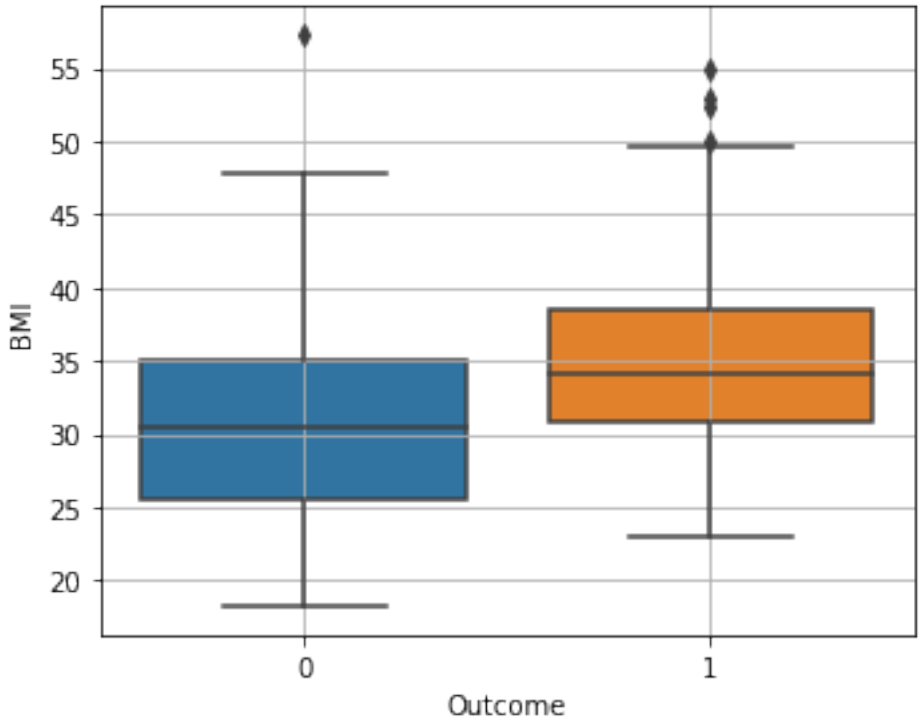
In [116]:
```
plt.figure(figsize=(5,4))
sns.boxplot(y=df['SkinThickness'],x=df['Outcome'])
sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
plt.tight_layout()
plt.grid(True)
plt.show()
```



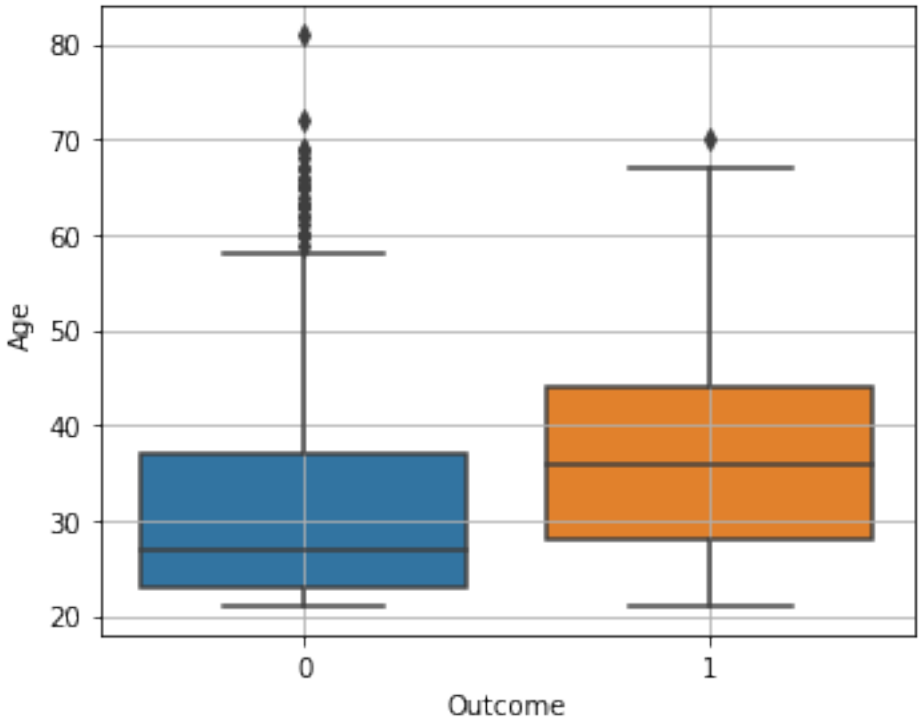
In [117]:
```
plt.figure(figsize=(5,4))
sns.boxplot(y=df['Insulin'],x=df['Outcome'])
sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
plt.tight_layout()
plt.grid(True)
plt.show()
```




```
In [118]: plt.figure(figsize=(5,4))
          sns.boxplot(y=df['BMI'],x=df['Outcome'])
          sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
          plt.tight_layout()
          plt.grid(True)
          plt.show()
```

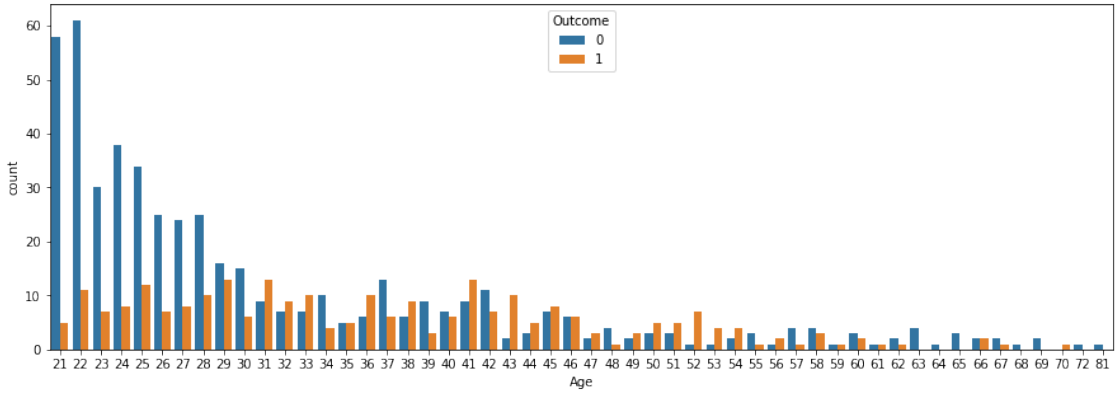



```
In [121]: plt.figure(figsize=(5,4))
          sns.boxplot(y=df['Age'],x=df['Outcome'])
          sns.color_palette("ch:start=.2,rot=-.3", as_cmap=True)
          plt.tight_layout()
          plt.grid(True)
          plt.show()
```




```
In [122]: plt.figure(figsize=(15,5))
          sns.countplot(x=df['Age'],hue=df['Outcome'])
```

```
Out[122]: <AxesSubplot:xlabel='Age', ylabel='count'>
```




```
In [123]: from sklearn.neighbors import KNeighborsClassifier
          from sklearn.linear_model import LogisticRegression
          from sklearn.svm import SVC
```

```
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import learning_curve

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import plot_roc_curve,roc_auc_score
from sklearn.metrics import confusion_matrix, classification_report,roc_auc_score ,ac
```

In [124]:
```
feature_columns = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insul

X = df[feature_columns]
y = df.Outcome
```

In [125]:
```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)

X_train.shape
```

Out[125]: (604, 7)

In [126]:
```
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
X_train_scaled = pd.DataFrame(ss.fit_transform(X_train))
X_test_scaled = pd.DataFrame(ss.transform(X_test))

X_train_scaled.columns = X_train.columns
X_test_scaled.columns = X_test.columns
```

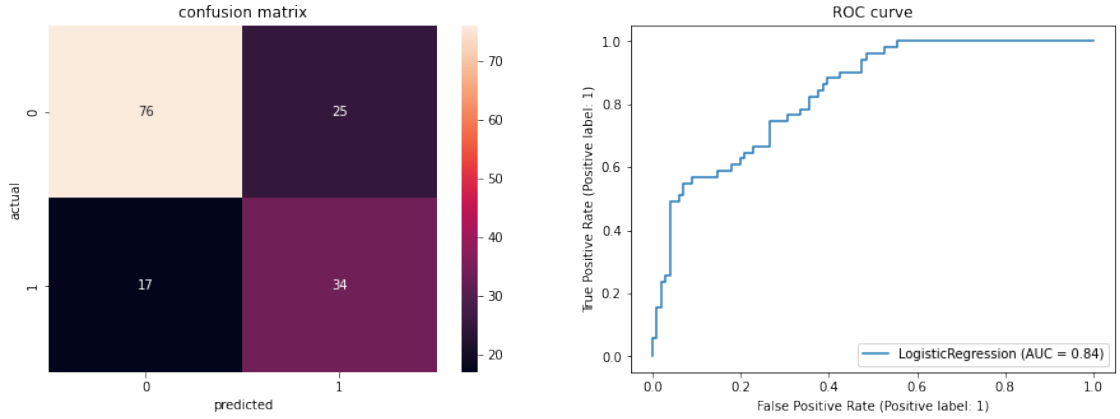
In [127]:
```
# Logistic Regression
model = LogisticRegression(class_weight = 'balanced',random_state = 42)
model.fit(X_train_scaled,y_train)
y_pred_lr = model.predict(X_test_scaled)
print(accuracy_score(y_test,y_pred_lr))
print(classification_report(y_test,y_pred_lr))
roc = roc_auc_score(y_test,y_pred_lr)
print('roc score : ',roc)
```

```
0.7236842105263158
              precision    recall  f1-score   support

           0       0.82      0.75      0.78       101
           1       0.58      0.67      0.62        51

    accuracy                           0.72       152
   macro avg       0.70      0.71      0.70       152
weighted avg       0.74      0.72      0.73       152
```

```
roc score :  0.7095709570957095
```

```
In [128]: fig,ax = plt.subplots(1,2,figsize=(15,5))
          matrix_lr = confusion_matrix(y_test, y_pred_lr)
          sns.heatmap(matrix_lr,annot=True,fmt='g',ax=ax[0])
          plot_roc_curve(model,X_test_scaled,y_test,ax=ax[1]);

          ax[0].title.set_text('confusion matrix')
          ax[1].title.set_text('ROC curve')

          ax[0].set_xlabel('predicted')
          ax[0].set_ylabel('actual');
```

```
/usr/local/lib/python3.8/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Functio
  warnings.warn(msg, category=FutureWarning)
```



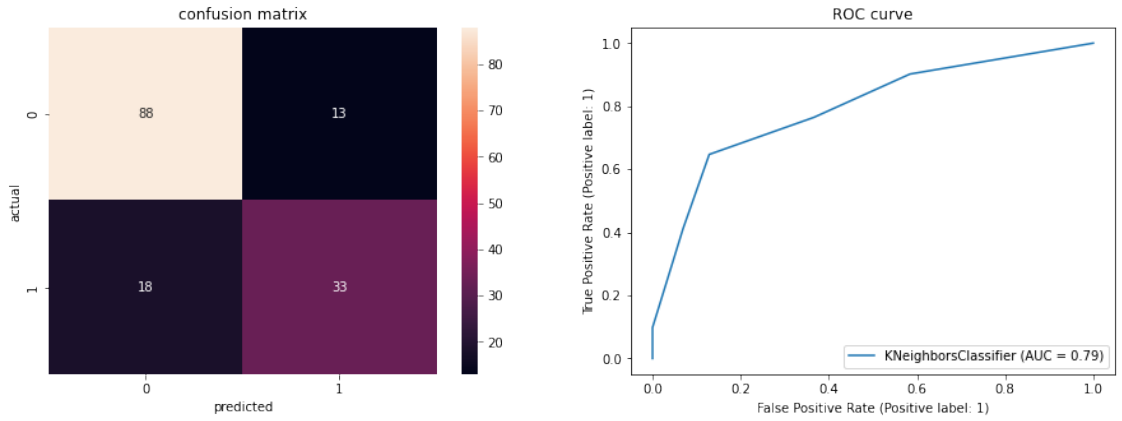

```
In [129]: # Knn
          knn = KNeighborsClassifier()
          model = knn
          model.fit(X_train_scaled,y_train)
          y_pred_knn = model.predict(X_test_scaled)
          print(accuracy_score(y_test,y_pred_knn))
          print(classification_report(y_test,y_pred_knn))
          roc = roc_auc_score(y_test,y_pred_knn)
          print('roc score : ',roc)
```

```
0.7960526315789473
              precision    recall  f1-score   support

           0       0.83      0.87      0.85       101
```

```
           1       0.72      0.65      0.68        51

    accuracy                           0.80       152
   macro avg       0.77      0.76      0.77       152
weighted avg       0.79      0.80      0.79       152

roc score :  0.7591729761211414


/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
```

```
In [130]: fig,ax = plt.subplots(1,2,figsize=(15,5))
          matrix_knn = confusion_matrix(y_test, y_pred_knn)
          sns.heatmap(matrix_knn,annot=True,fmt='g',ax=ax[0])
          plot_roc_curve(model,X_test_scaled,y_test,ax=ax[1]);

          ax[0].title.set_text('confusion matrix')
          ax[1].title.set_text('ROC curve')

          ax[0].set_xlabel('predicted')
          ax[0].set_ylabel('actual');
```

```
/usr/local/lib/python3.8/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Functio
  warnings.warn(msg, category=FutureWarning)
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
```



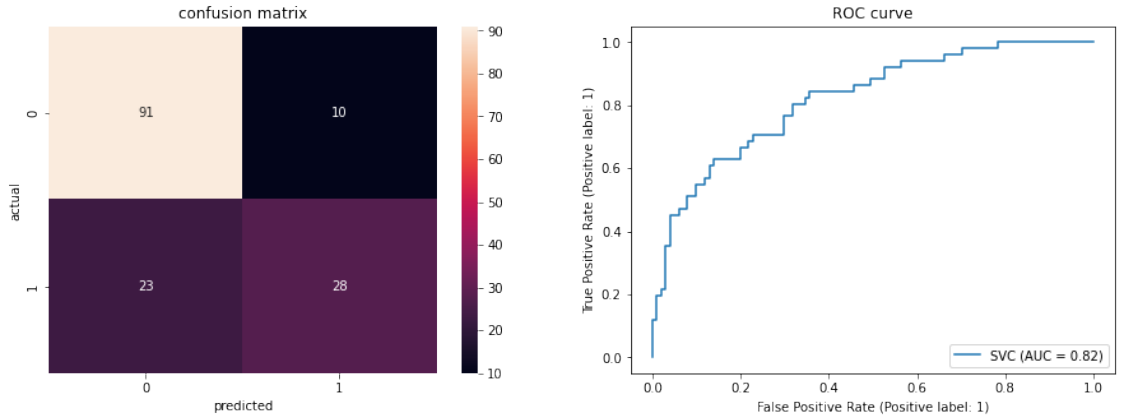

```
In [131]: sv = SVC()
          model = sv
          model.fit(X_train_scaled,y_train)
```

```
y_pred_sv = model.predict(X_test_scaled)
print(accuracy_score(y_test,y_pred_sv))
print(classification_report(y_test,y_pred_sv))
roc = roc_auc_score(y_test,y_pred_knn)
print('roc score : ',roc)
```

```
0.7828947368421053
              precision    recall  f1-score   support

           0       0.80      0.90      0.85       101
           1       0.74      0.55      0.63        51

    accuracy                           0.78       152
   macro avg       0.77      0.73      0.74       152
weighted avg       0.78      0.78      0.77       152

roc score :  0.7591729761211414
```

```
In [132]: fig,ax = plt.subplots(1,2,figsize=(15,5))
          matrix_sv = confusion_matrix(y_test, y_pred_sv)
          sns.heatmap(matrix_sv,annot=True,fmt='g',ax=ax[0])
          plot_roc_curve(model,X_test_scaled,y_test,ax=ax[1]);

          ax[0].title.set_text('confusion matrix')
          ax[1].title.set_text('ROC curve')

          ax[0].set_xlabel('predicted')
          ax[0].set_ylabel('actual');
```

```
/usr/local/lib/python3.8/site-packages/sklearn/utils/deprecation.py:87: FutureWarning: Functio
  warnings.warn(msg, category=FutureWarning)
```



```
In [ ]: #Tuning by Grid
```

```
In [133]: def grid_evaluation(model, X_train_scaled, y_train, X_test_scaled, y_test):
              model.fit(X_train_scaled, y_train)
              y_pred = model.predict(X_test_scaled)
              print(f"Best Parameters : {model.best_params_}")
              print("*"*50)
              # Best Score: Mean cross-validated score of the best_estimator
              print(f"\nBest Score :  {model.best_score_}")
              print("*"*50)
              print(f"\nAccuracy Score (Train set) :  {model.score(X_train_scaled,y_train)}")
              print("*"*50)
              print(f"\nAccuracy Score (Test set): {accuracy_score(y_test, y_pred)}")
              print("*"*50)
              print(f"\nRoc auc score : {roc_auc_score(y_test, y_pred)}")
              print("*"*50)
              print(f"\nConfusion Matrix : \n {confusion_matrix(y_test, y_pred)}")
              print("*"*50)
              print(f"\nClassification Report : \n {classification_report(y_test, y_pred)}")
```

```
In [134]: # LogisticRegression tuned
          param_grid = {'C': [1,5,10]}

          best_param_lr = {'C':[10]}

          lrt = LogisticRegression()
          grid_lrt = GridSearchCV(lrt, best_param_lr, scoring='accuracy', refit=True)
          grid_evaluation(grid_lrt, X_train_scaled, y_train, X_test_scaled, y_test)
```

```
Best Parameters : {'C': 10}
**************************************************

Best Score :  0.7648898071625345
**************************************************

Accuracy Score (Train set) :  0.7649006622516556
**************************************************

Accuracy Score (Test set): 0.7960526315789473
**************************************************

Roc auc score : 0.7397592700446515
**************************************************

Confusion Matrix :
 [[92  9]
 [22 29]]
```

```
************************************************

Classification Report :
              precision    recall  f1-score   support

           0       0.81      0.91      0.86       101
           1       0.76      0.57      0.65        51

    accuracy                           0.80       152
   macro avg       0.79      0.74      0.75       152
weighted avg       0.79      0.80      0.79       152
```

```
In [136]: # KNN tuned
          param_grid = {'n_neighbors' : np.arange(1, 30, 2),
                        'metric' : ['euclidean', 'minkowski', 'manhatten']}

          best_param_knn = {'metric': ['euclidean'], 'n_neighbors': [25]}

          knnt = KNeighborsClassifier()
          grid_knnt = GridSearchCV(knnt, best_param_knn, scoring='accuracy', cv=10, refit=True)
          grid_evaluation(grid_knnt, X_train_scaled, y_train, X_test_scaled, y_test)
```

```
Best Parameters : {'metric': 'euclidean', 'n_neighbors': 25}
************************************************

Best Score :  0.7549726775956285
************************************************

Accuracy Score (Train set) :  0.7764900662251656
************************************************

Accuracy Score (Test set): 0.75
************************************************

Roc auc score : 0.6856920986216268
************************************************

Confusion Matrix :
 [[89 12]
 [26 25]]
************************************************

Classification Report :
              precision    recall  f1-score   support

           0       0.77      0.88      0.82       101
```

```
           1       0.68      0.49      0.57        51

    accuracy                           0.75       152
   macro avg       0.72      0.69      0.70       152
weighted avg       0.74      0.75      0.74       152



/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
/usr/local/lib/python3.8/site-packages/sklearn/base.py:441: UserWarning: X does not have valid
  warnings.warn(
```

In [137]:
```
#  svc tuned
param_grid = {'C': [0.01, 0.1, 1, 10, 100],
              'gamma': [1,0.1,0.01,0.001],
              'kernel': ['rbf']}

best_param_svc = [{'C': [100], 'gamma': [0.001], 'kernel': ['rbf']}]

svct = SVC()
grid_svct = GridSearchCV(svct, best_param_svc, scoring='accuracy', cv=10, refit=True
grid_evaluation(grid_svct, X_train_scaled, y_train, X_test_scaled, y_test)
```

```
Best Parameters : {'C': 100, 'gamma': 0.001, 'kernel': 'rbf'}
**************************************************
```

```
Best Score :  0.7515846994535519
**************************************************

Accuracy Score (Train set) :  0.7615894039735099
**************************************************

Accuracy Score (Test set): 0.8026315789473685
**************************************************

Roc auc score : 0.7447097650941565
**************************************************

Confusion Matrix :
 [[93  8]
 [22 29]]
**************************************************

Classification Report :
               precision    recall  f1-score   support

           0       0.81      0.92      0.86       101
           1       0.78      0.57      0.66        51

    accuracy                           0.80       152
   macro avg       0.80      0.74      0.76       152
weighted avg       0.80      0.80      0.79       152


In [ ]:
```