In [3]:
```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen

Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving diabetes.csv to diabetes.csv
```

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
from sklearn import tree
import warnings
warnings.filterwarnings('ignore')
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecat
ed. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

In [5]:
```
df = pd.read_csv("diabetes.csv")
```

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

Out[6]:

	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

meaning of columns :

In [7]:
```
#Pregnancies :Number of times pregnant
#Glucose :Plasma glucose concentration a 2 hours in an oral glucose tolerance test
#BloodPressure :Diastolic blood pressure (mm Hg)
#SkinThickness :Triceps skin fold thickness (mm)
#Insulin :2-Hour serum insulin (mu U/ml)
#BMI :Body mass index (weight in kg/(height in m)^2)
#DiabetesPedigreeFunction :Diabetes pedigree function
#Age :Age (years)
#Outcome :Class variable (0 or 1) 268 of 768 are 1, the others are 0
```

In [8]:
```
df.describe(include="all")
```

Out[8]:

	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

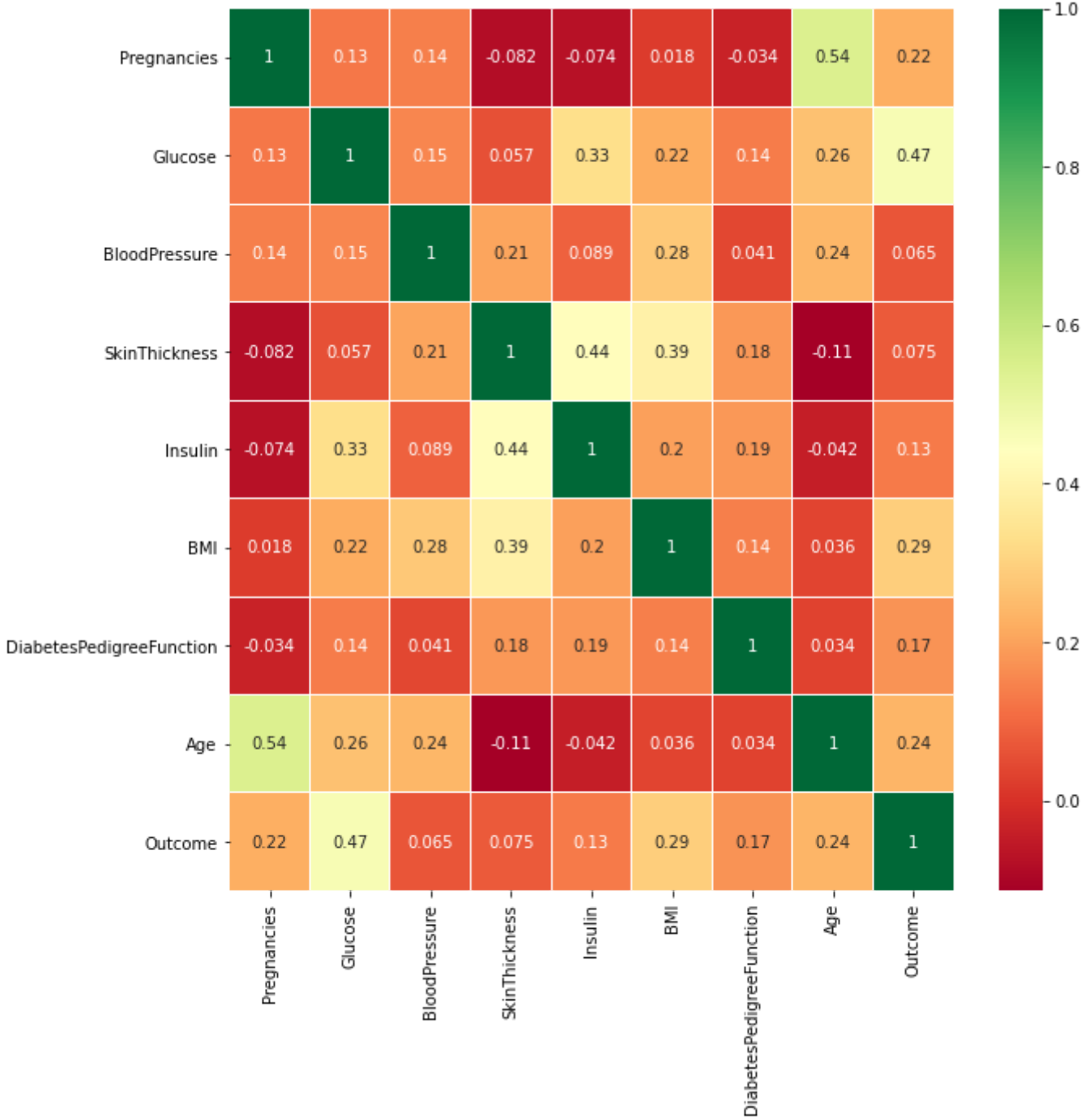
In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

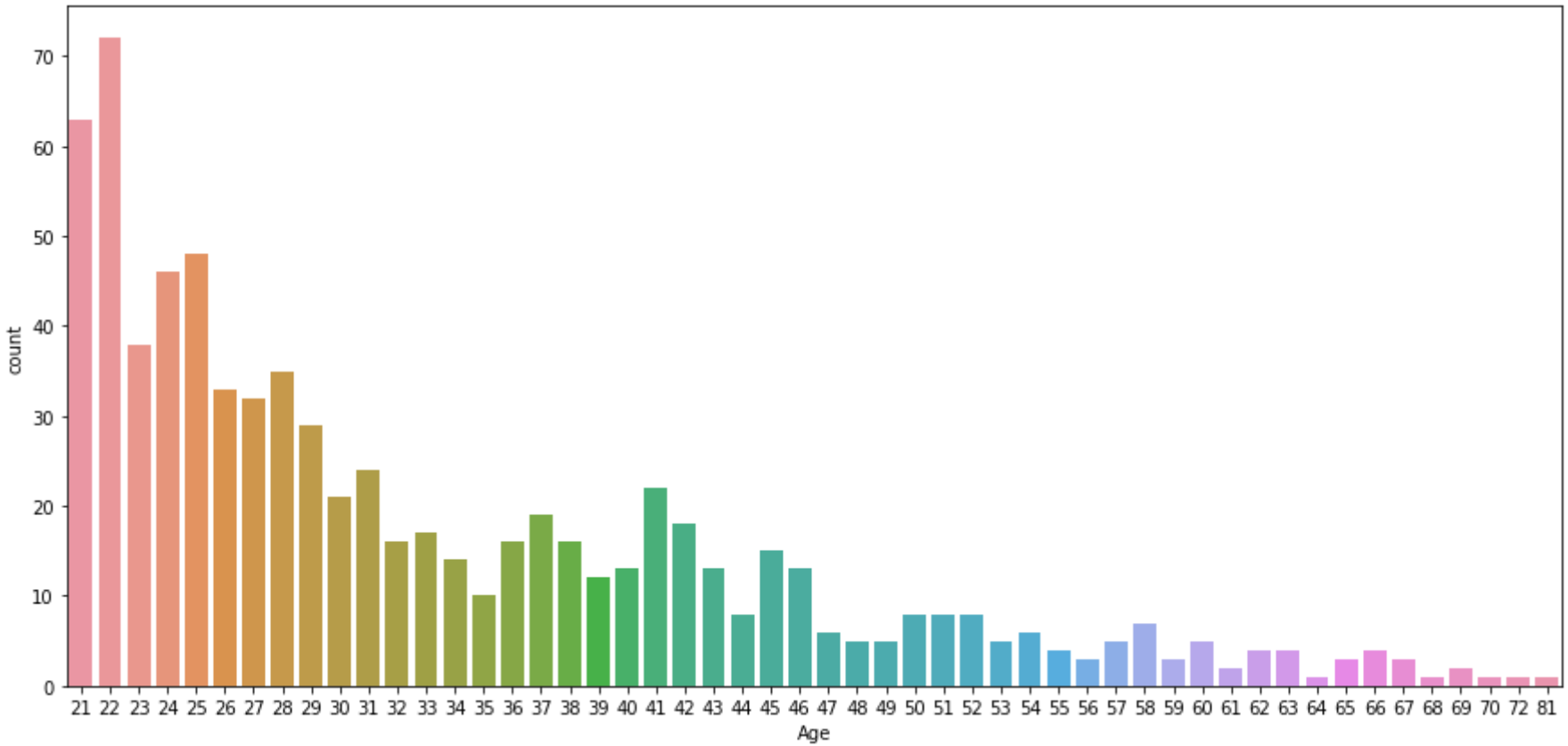
In [10]:
```
df.info()
```

```
<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
```

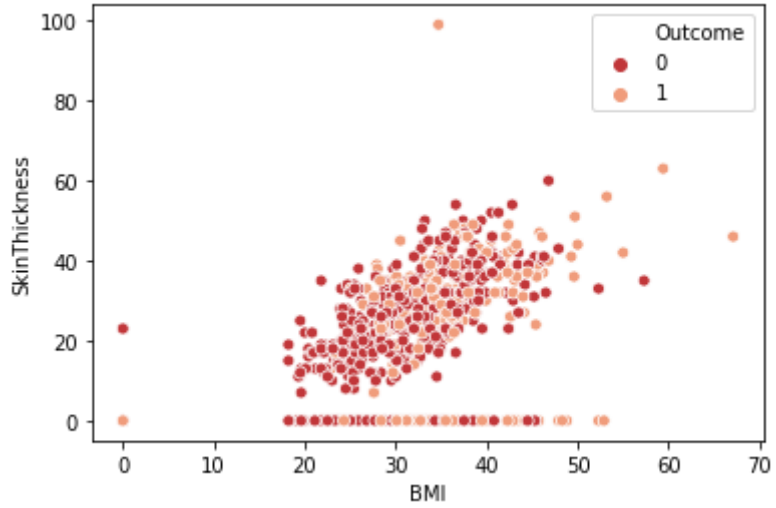
In [11]:
```
df.corr()
sns.heatmap(df.corr(),annot=True,cmap='RdYlGn',linewidths=0.2)
fig=plt.gcf()
fig.set_size_inches(10,10)
plt.show()
```


Pregnancies 1 0.13 0.14 -0.082 -0.074 0.018 -0.034 0.54 0.22
Glucose 0.13 1 0.15 0.057 0.33 0.22 0.14 0.26 0.47
BloodPressure 0.14 0.15 1 0.21 0.089 0.28 0.041 0.24 0.065
SkinThickness -0.082 0.057 0.21 1 0.44 0.39 0.18 -0.11 0.075
Insulin -0.074 0.33 0.089 0.44 1 0.2 0.19 -0.042 0.13
BMI 0.018 0.22 0.28 0.39 0.2 1 0.14 0.036 0.29
DiabetesPedigreeFunction -0.034 0.14 0.041 0.18 0.19 0.14 1 0.034 0.17
Age 0.54 0.26 0.24 -0.11 -0.042 0.036 0.034 1 0.24
Outcome 0.22 0.47 0.065 0.075 0.13 0.29 0.17 0.24 1
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
1.0
0.8
0.6
0.4
0.2
0.0

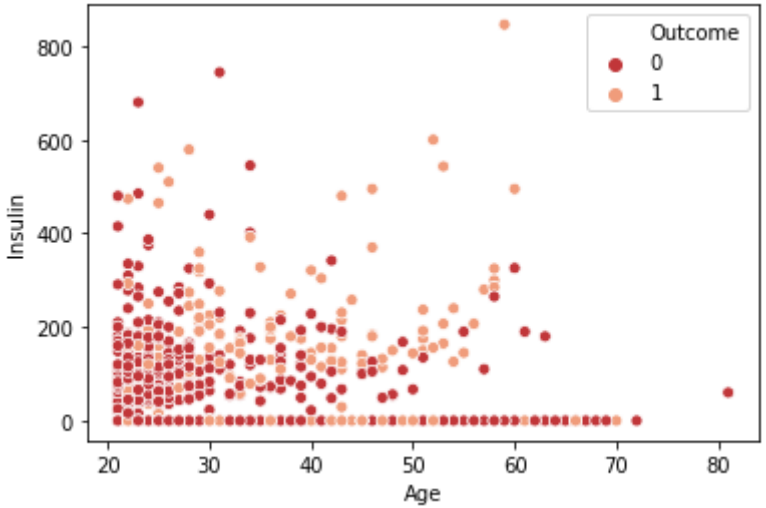
In [12]:
```
plt.figure(figsize=(15,7))
sns.set_palette("RdBu")
sns.countplot(x="Age", data=df)
plt.show()
```



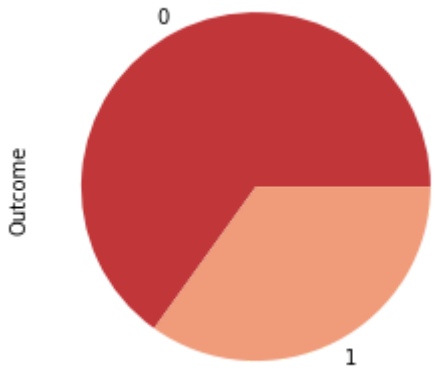
In [13]:
```
sns.scatterplot(x="BMI", y="SkinThickness",data=df, hue="Outcome")
plt.show()
```



In [14]:
```
sns.scatterplot(x="Age", y="Insulin",data=df, hue="Outcome")
plt.show()
```

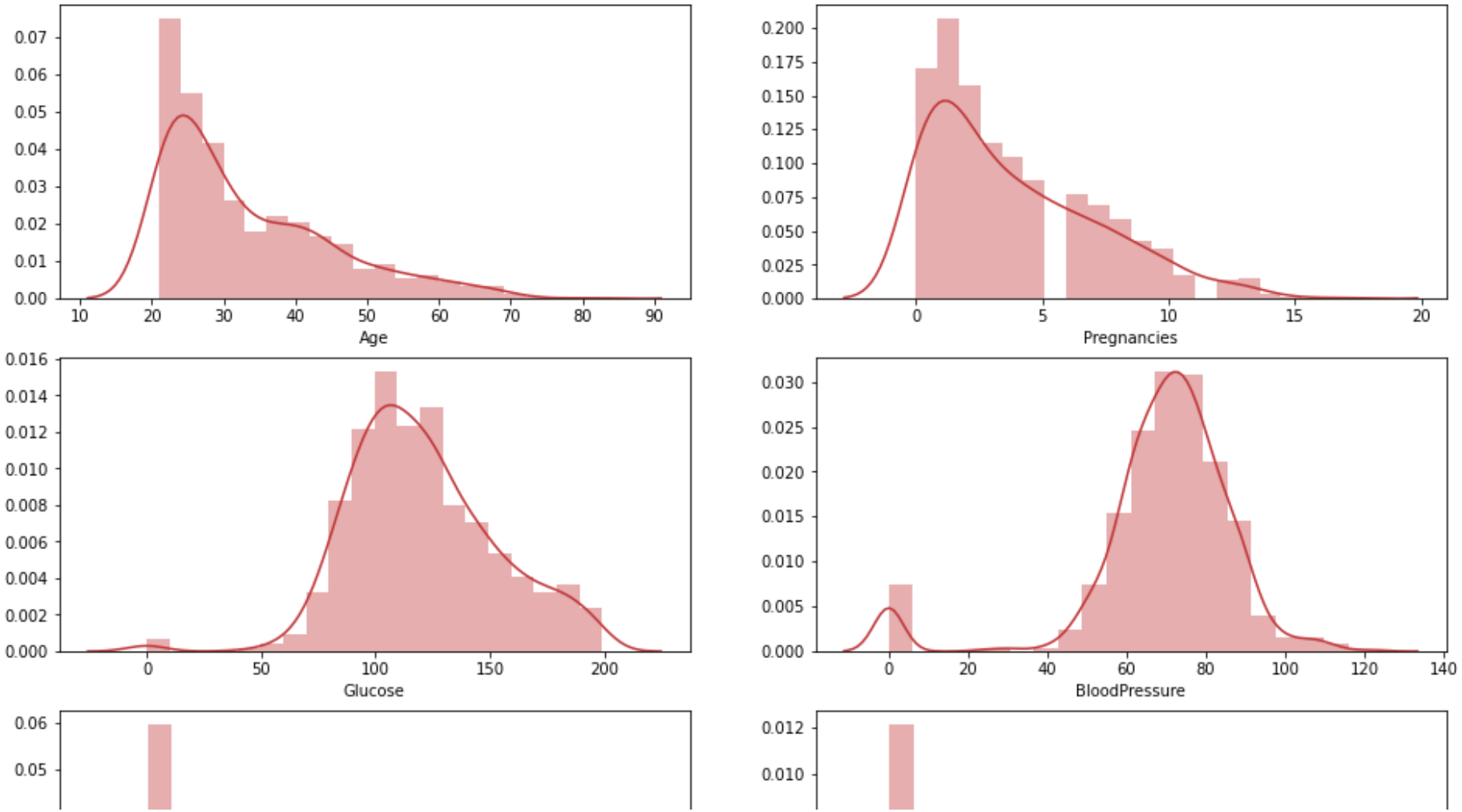


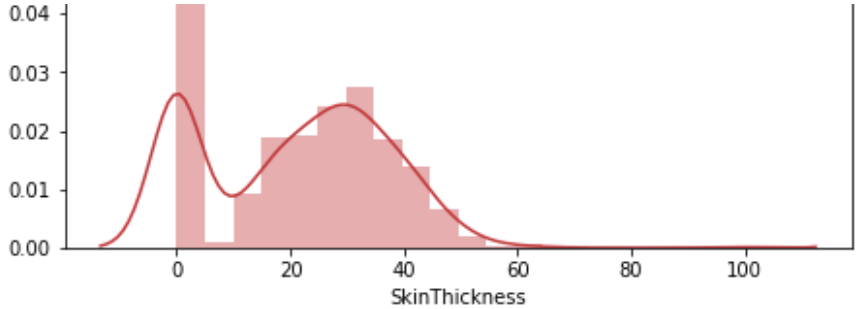
In [15]:
```
df["Outcome"].value_counts().plot(kind="pie")
plt.show()
```

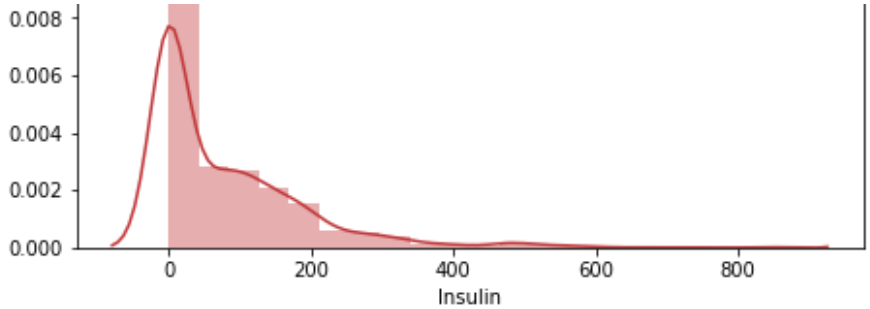


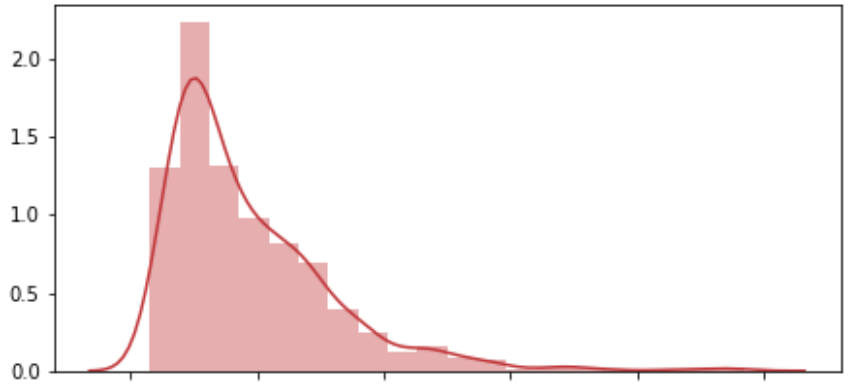
In [16]:
```
fig, ax = plt.subplots(4,2, figsize=(16,16))
sns.distplot(df.Age, bins = 20, ax=ax[0,0])
sns.distplot(df.Pregnancies, bins = 20, ax=ax[0,1])
sns.distplot(df.Glucose, bins = 20, ax=ax[1,0])
sns.distplot(df.BloodPressure, bins = 20, ax=ax[1,1])
sns.distplot(df.SkinThickness, bins = 20, ax=ax[2,0])
sns.distplot(df.Insulin, bins = 20, ax=ax[2,1])
sns.distplot(df.DiabetesPedigreeFunction, bins = 20, ax=ax[3,0])
sns.distplot(df.BMI, bins = 20, ax=ax[3,1])
```

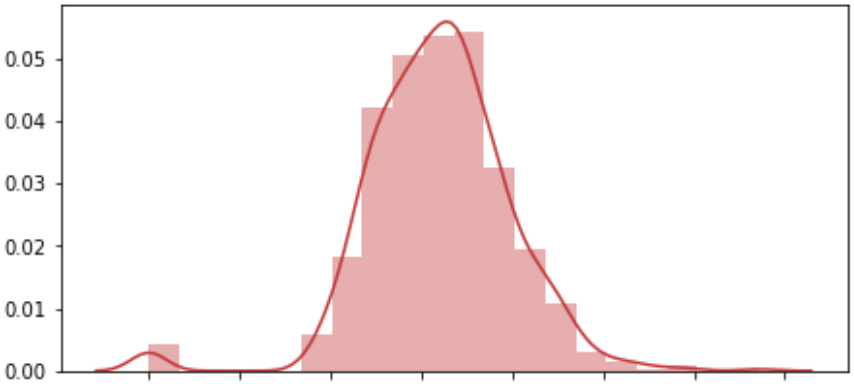
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f8ee7a73b70>











In [17]:
```
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
```

In [18]:
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,random_state=1)
```

Ada Boosting :

In [19]:
```
from sklearn.ensemble import AdaBoostClassifier
```

In [20]:
```
ada = AdaBoostClassifier(n_estimators=100)
ada.fit(X_train,y_train)
```

Out[20]:
```
AdaBoostClassifier(algorithm='SAMME.R', base_estimator=None, learning_rate=1.0,
                   n_estimators=100, random_state=None)
```

In [21]:
```
y_pred = ada.predict(X_test)
```

In [22]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.88      0.84       146
           1       0.76      0.66      0.70        85

    accuracy                           0.80       231
   macro avg       0.79      0.77      0.77       231
weighted avg       0.79      0.80      0.79       231
```

Gradient Boosting :

In [23]:
```
from sklearn.ensemble import GradientBoostingClassifier
```

In [24]:
```
gb = GradientBoostingClassifier(n_estimators=100)
gb.fit(X_train,y_train)
```

Out[24]:
```
GradientBoostingClassifier(ccp_alpha=0.0, criterion='friedman_mse', init=None,
                           learning_rate=0.1, loss='deviance', max_depth=3,
                           max_features=None, max_leaf_nodes=None,
                           min_impurity_decrease=0.0, min_impurity_split=None,
                           min_samples_leaf=1, min_samples_split=2,
                           min_weight_fraction_leaf=0.0, n_estimators=100,
                           n_iter_no_change=None, presort='deprecated',
                           random_state=None, subsample=1.0, tol=0.0001,
                           validation_fraction=0.1, verbose=0,
                           warm_start=False)
```

In [25]:
```
y_pred = gb.predict(X_test)
```

In [26]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.88      0.85       146
           1       0.76      0.65      0.70        85

    accuracy                           0.80       231
   macro avg       0.79      0.77      0.77       231
weighted avg       0.79      0.80      0.79       231
```

XG Boosting :

In [27]:
```
from xgboost import XGBClassifier
```

In [28]:
```
xgb = XGBClassifier(n_estimators=200,reg_alpha=1)
```

In [29]:
```
xgb.fit(X_train,y_train)
```

Out[29]:
```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=200, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=1, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

In [30]:
```
y_pred = xgb.predict(X_test)
```

In [31]:
```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.89      0.86       146
           1       0.78      0.67      0.72        85

    accuracy                           0.81       231
   macro avg       0.80      0.78      0.79       231
weighted avg       0.81      0.81      0.81       231
```

In [ ]:
```

```