Diabetes Patients Predictive Analysis Project

Project Overview

- This project focuses on predictive analysis for diabetes diagnosis using a dataset originally sourced from the National Institute of Diabetes and Digestive and Kidney Diseases.
- The dataset contains various medical and demographic variables for a group of Pima Indian heritage females who are at least 21 years old.
- The primary objective of this project is to develop a predictive model that can diagnostically predict whether a patient has diabetes based on the provided diagnostic measurements and demographic information.

Features And Description

- 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)

IMPORTING PACKAGES | LIBRARIES

In [6]:
```
import numpy as np
import pandas as pd
from pandas.plotting import scatter_matrix
```

```
import matplotlib.pyplot as plt
import seaborn as sns
import missingno as msno
import plotly.graph_objs as go
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, accuracy_score,confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report
```

READING DATA FROM CSV FILE

In [9]:
```
pwd # Current Working Directory
```

Out[9]:
```
'C:\\Users\\hkuma\\hemanth\\Portfolio Projects\\Diabetes Project'
```

In [11]:
```
Diabetes = pd.read_csv("Diabetes.csv")
Diabetes.head()
```

Out[11]:

	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

EXPLORATORY DATA ANALYSIS

In [14]:
```
# Total no of columns in Dataset
Diabetes.columns
```

Out[14]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [16]:
```
# Information About Dataset
Diabetes.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 [18]:
```
# More About Dataset
Diabetes.describe()
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	7(
count	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	
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	

In [20]:
```
# More About Dataset with Transpose ('T')
Diabetes.describe().T
```

Out[20]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25000	199.00
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00000	122.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00000	99.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25000	846.00
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

In [22]:
```
# Checking for null values in Dataset
Diabetes.isnull()
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
763	False	False	False	False	False	False	False	False	False
764	False	False	False	False	False	False	False	False	False
765	False	False	False	False	False	False	False	False	False
766	False	False	False	False	False	False	False	False	False
767	False	False	False	False	False	False	False	False	False

768 rows × 9 columns

In [24]:
```
# Checking Total null values in Dataset
Diabetes.isnull().sum()
```

Out[24]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [28]:
```
Diabetes_copy = Diabetes.copy
Diabetes_copy = Diabetes_copy (deep= True)
Diabetes_copy[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI']]
```

Out[28]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI
0	148	72	35	0	33.6
1	85	66	29	0	26.6
2	183	64	0	0	23.3
3	89	66	23	94	28.1
4	137	40	35	168	43.1
...	...	...	...	...	...
763	101	76	48	180	32.9
764	122	70	27	0	36.8
765	121	72	23	112	26.2
766	126	60	0	0	30.1
767	93	70	31	0	30.4

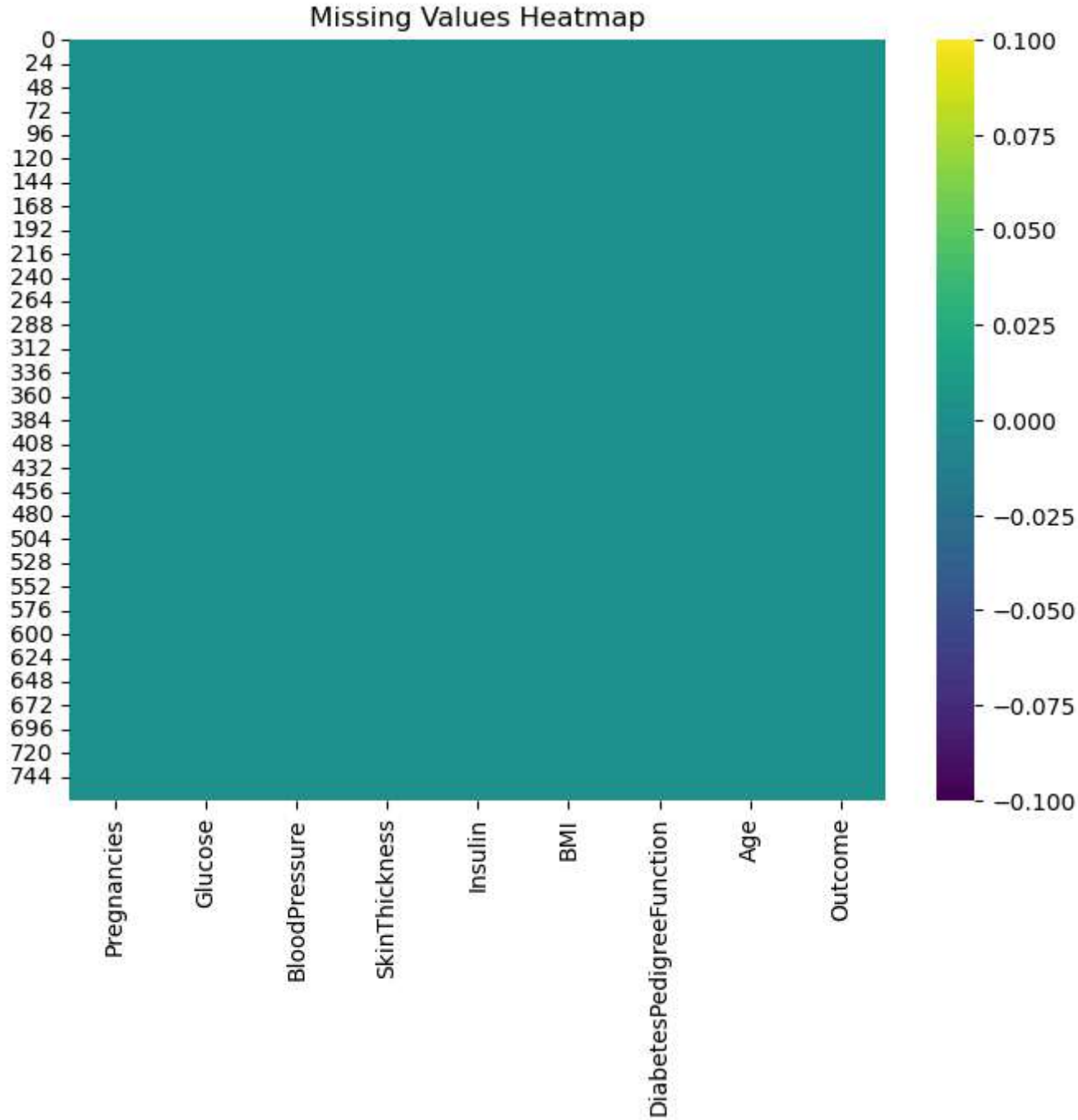
768 rows × 5 columns

In [30]:
```
print(Diabetes.isnull().sum())
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

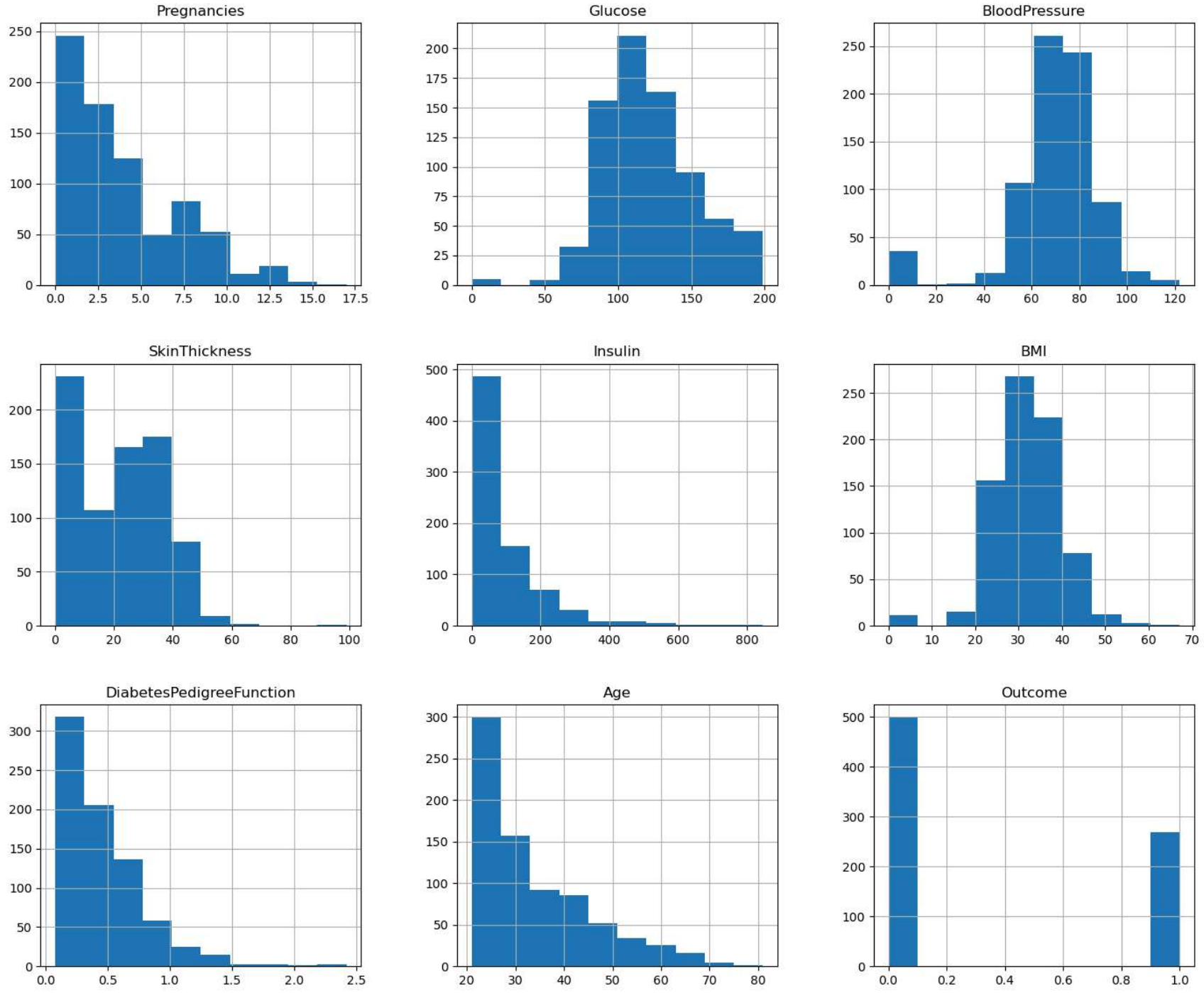
Heatmap to Check Missing Values in Dataset

In [37]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(Diabetes.isnull(), cmap='viridis')
plt.title('Missing Values Heatmap')
plt.show()
```


Missing Values Heatmap
0
24
48
72
96
120
144
168
192
216
240
264
288
312
336
360
384
408
432
456
480
504
528
552
576
600
624
648
672
696
720
744
Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100

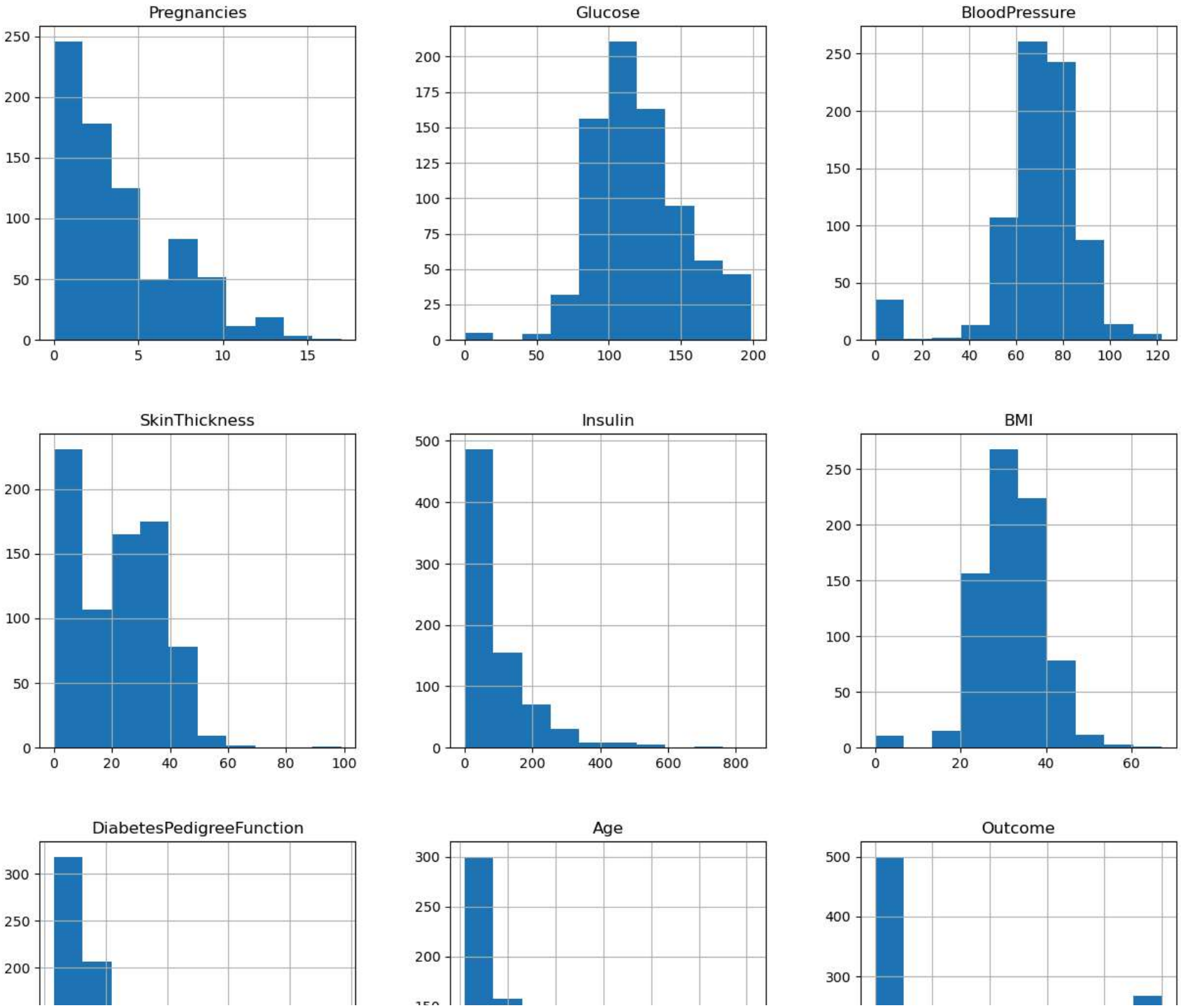
DATA VISUALIZATION

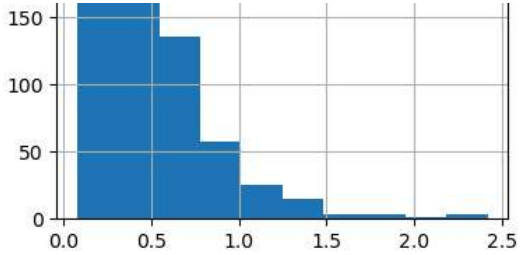
In [13]:
```
# Plotting The Data Distribution Plots
Diabetes.hist(figsize = (17,14))
plt.show()
```

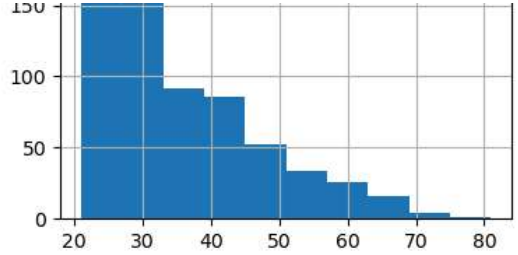

Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome

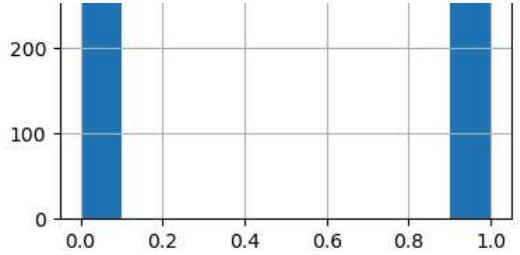
In [14]:
```
# Aiming to input NAN values for the column in accordance with their distribution
Diabetes_copy['Glucose'].fillna(Diabetes_copy['Glucose'].mean(), inplace=True)
Diabetes_copy['BloodPressure'].fillna(Diabetes['BloodPressure'].mean(), inplace=True)
Diabetes_copy['SkinThickness'].fillna(Diabetes['SkinThickness'].median(), inplace=True)
Diabetes_copy['Insulin'].fillna (Diabetes['Insulin'].median(), inplace= True)
Diabetes_copy['BMI'].fillna(Diabetes['BMI'].median(), inplace= True)
```

In [15]:
```
# Plotting The Data Distribution Plots After removing NaN values from Dataset
P = Diabetes.hist(figsize=(15,15))
```

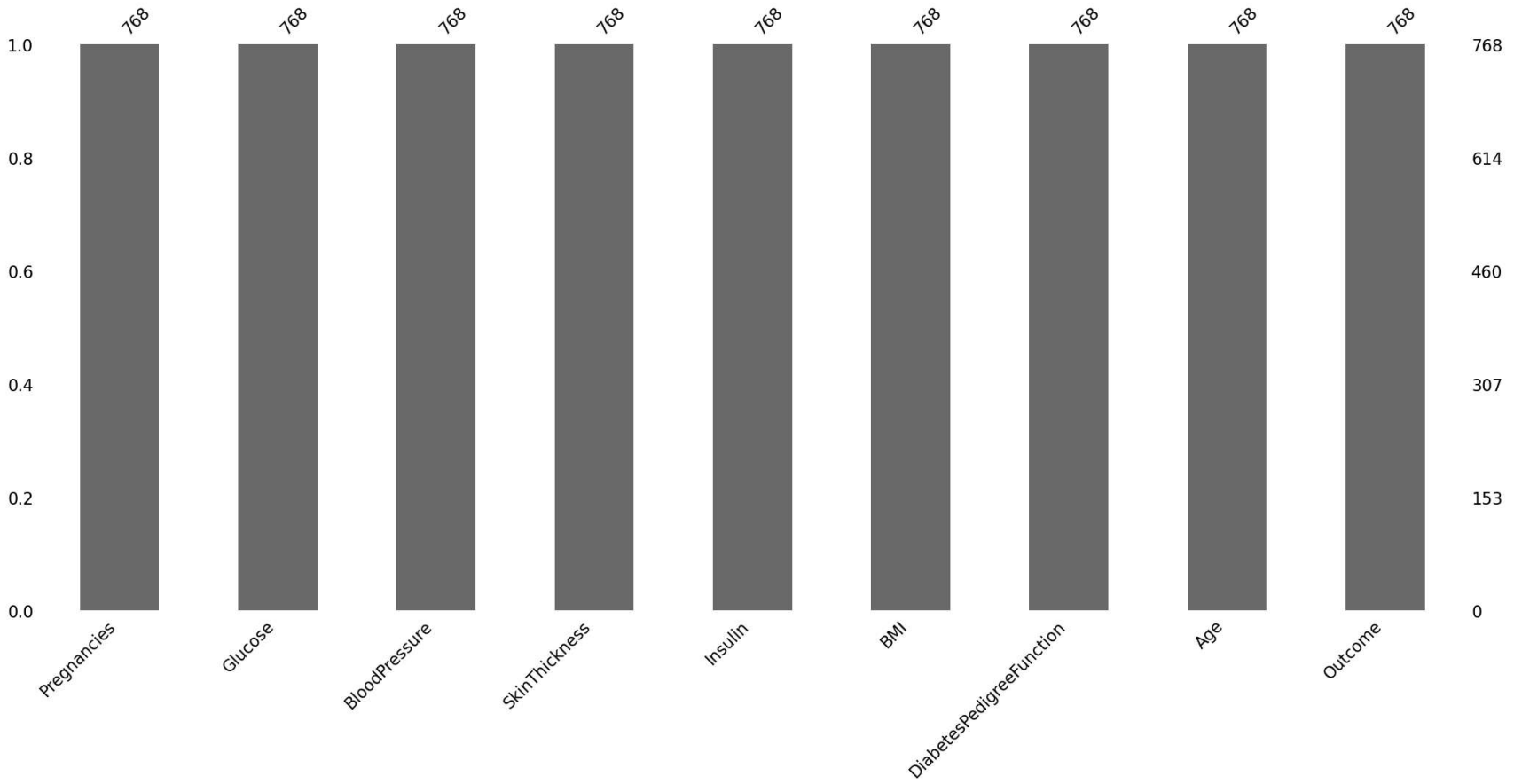

Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome







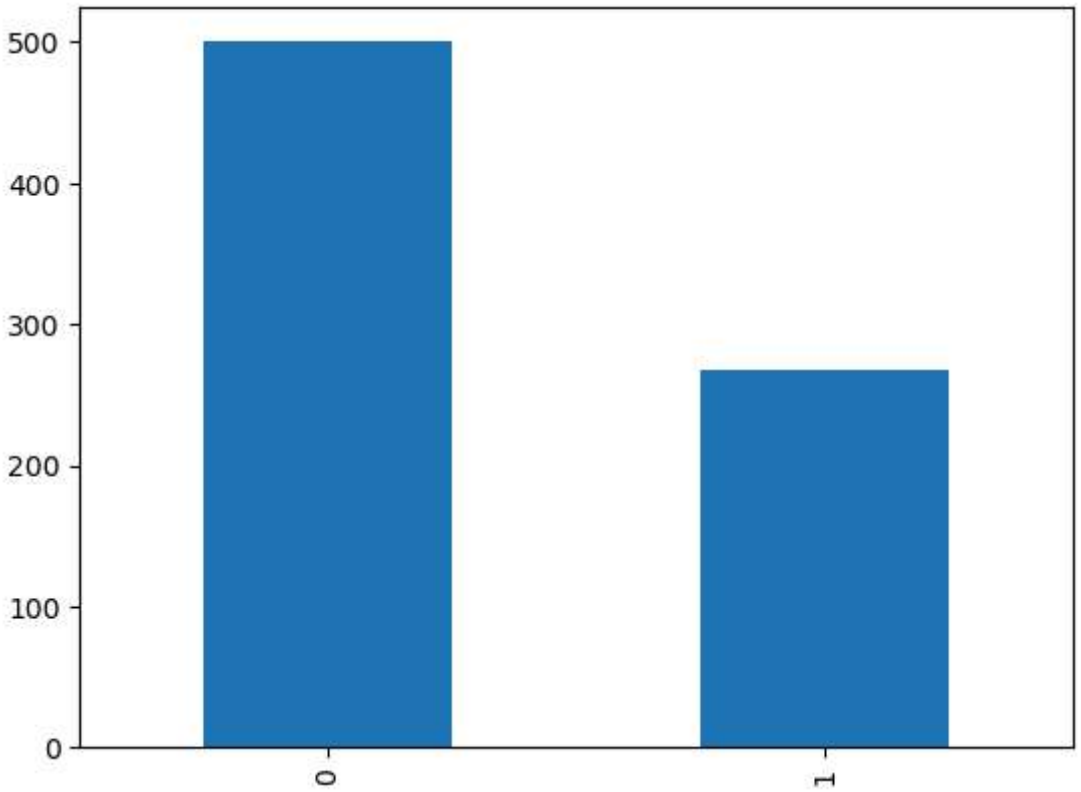
In [16]:
```
# Plotting NaN Count analysis Plot
P = msno.bar(Diabetes)
```



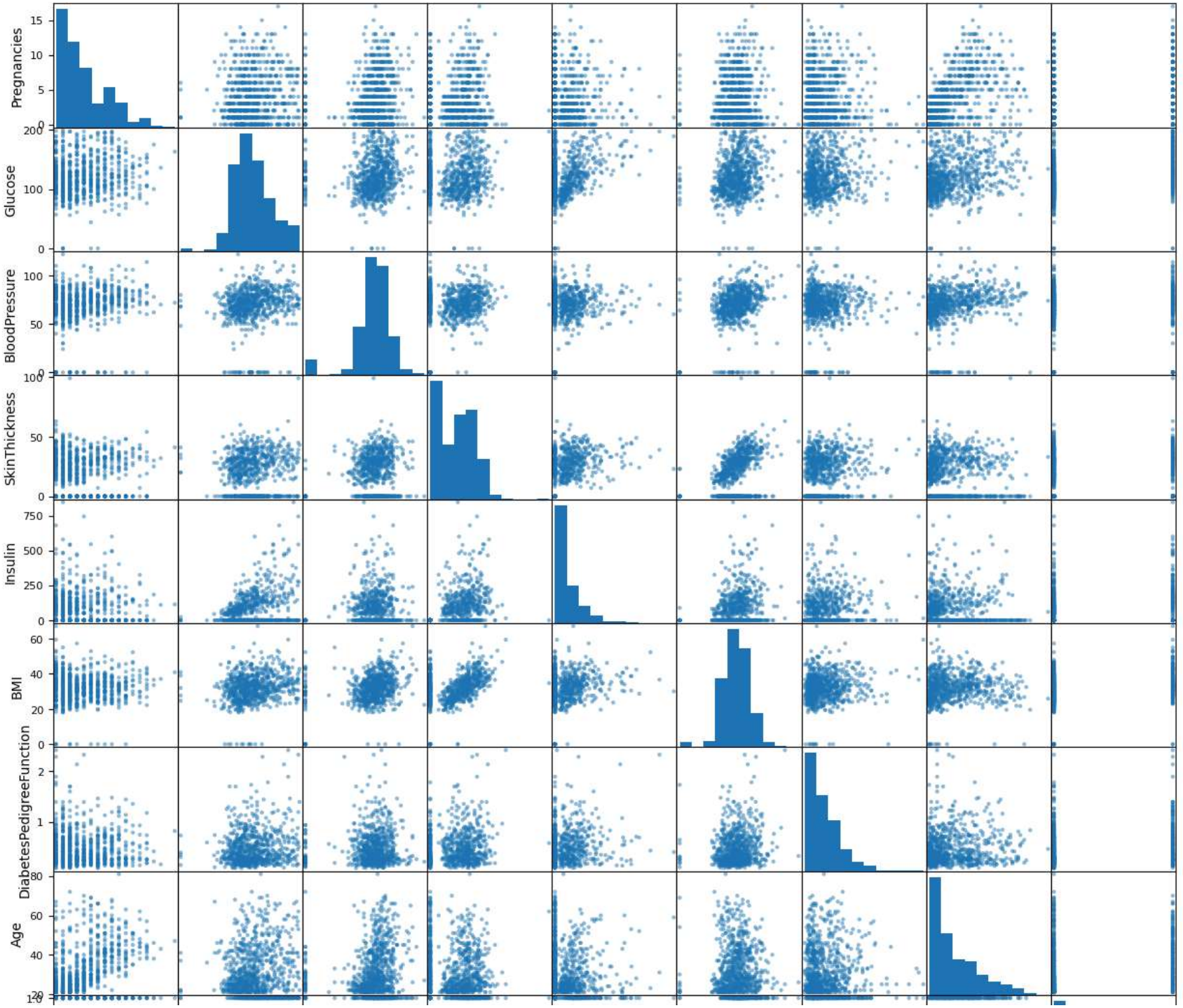
In [17]:
```
# Checking the balance of the data by plotting the count of outcomes by their value
color_wheel={1: "#0392cf", 2: "#7bc043"}
colors= Diabetes["Outcome"].map(lambda x:color_wheel.get(x+1))
```

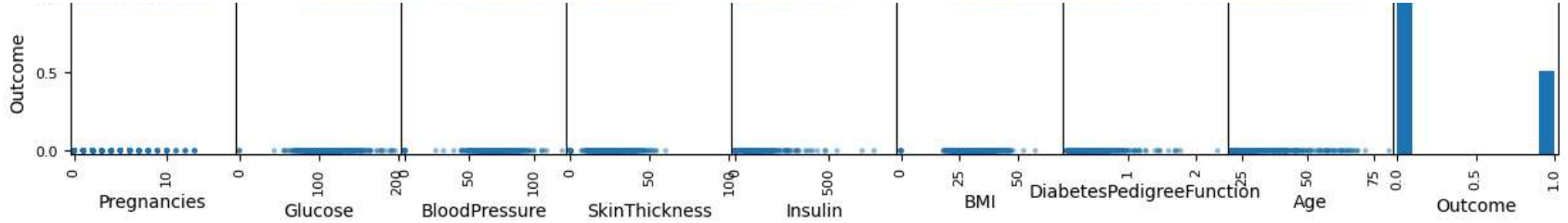
```
print(Diabetes.Outcome.value_counts())
p = Diabetes.Outcome.value_counts().plot(kind="bar")
```

```
0    500
1    268
Name: Outcome, dtype: int64
```

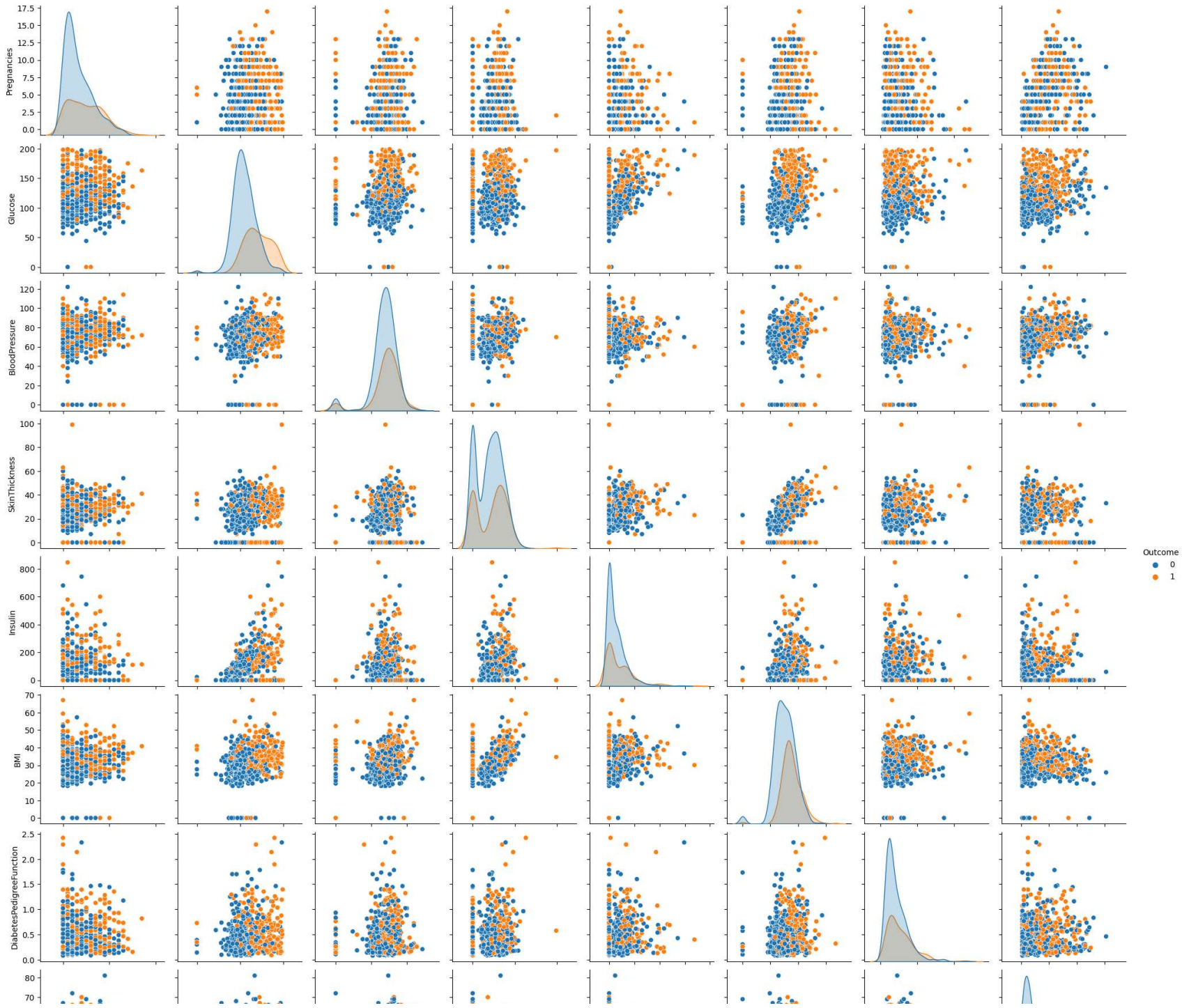


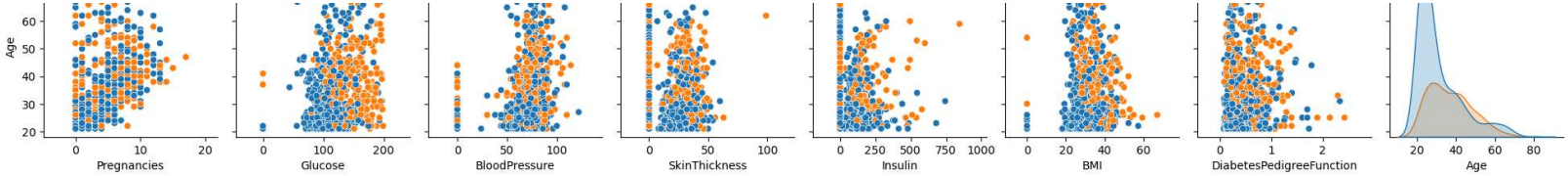
In [18]:
```
# Plot Scatter Matrix of uncleaned Data
P = scatter_matrix(Diabetes,figsize = (15,15))
plt.show()
```


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age

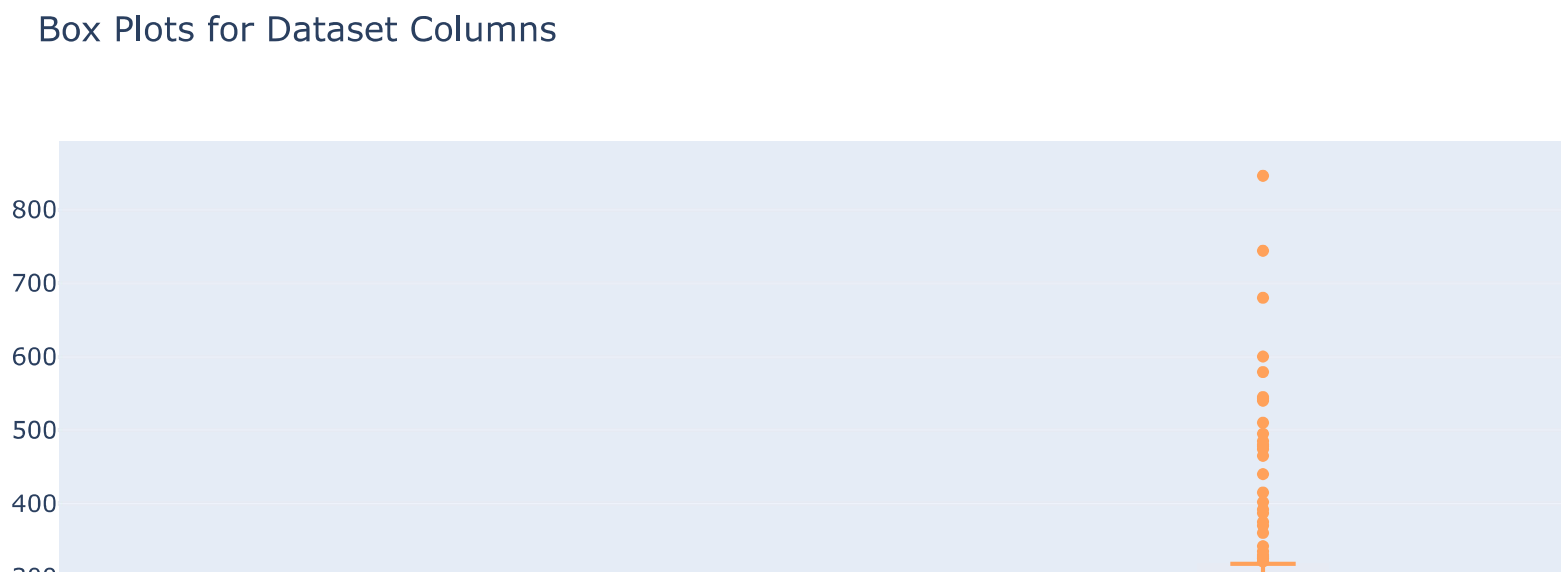


In [19]:
```
# Plotting Pair Plot for the Dataset
sns.pairplot(Diabetes_copy,hue='Outcome')
plt.show()
```


Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Outcome
0
1

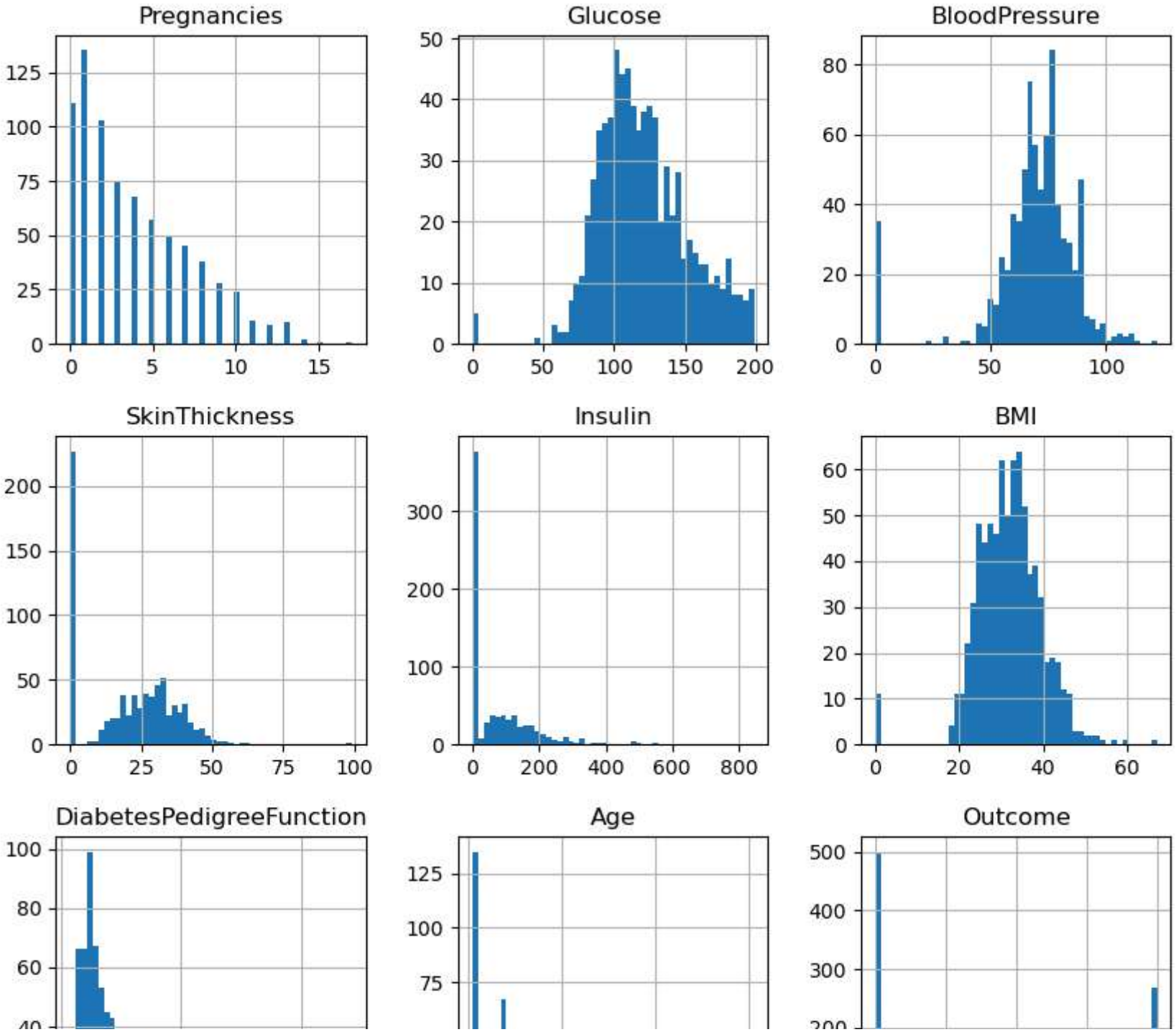


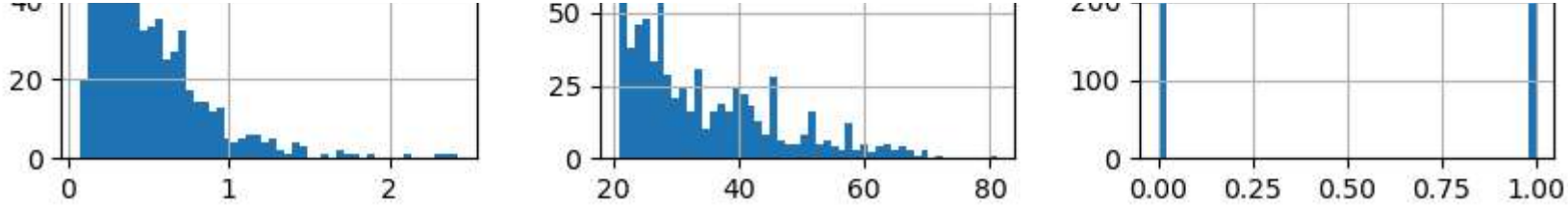
In [20]:
```
# Create a list to store the box plot traces
box_traces = []

# Iterate through each column and create a box plot
for column in Diabetes.columns:
    if column != 'Outcome': # Exclude 'Outcome' if it's the target variable
        trace = go.Box(y=Diabetes[column], name=column)
        box_traces.append(trace)

# Create a layout
layout = go.Layout(title='Box Plots for Dataset Columns')

# Create a figure and add the traces and layout
fig = go.Figure(data=box_traces, layout=layout)

# Show the figure
fig.show()
```



In [21]:
```
# Plot the histogram
fig, ax = plt.subplots(figsize=(10, 10))
Diabetes.hist(bins=50, ax=ax)
plt.show()
```

```
C:\Users\prata\AppData\Local\Temp\ipykernel_15472\3315462812.py:3: UserWarning:

To output multiple subplots, the figure containing the passed axes is being cleared.
```

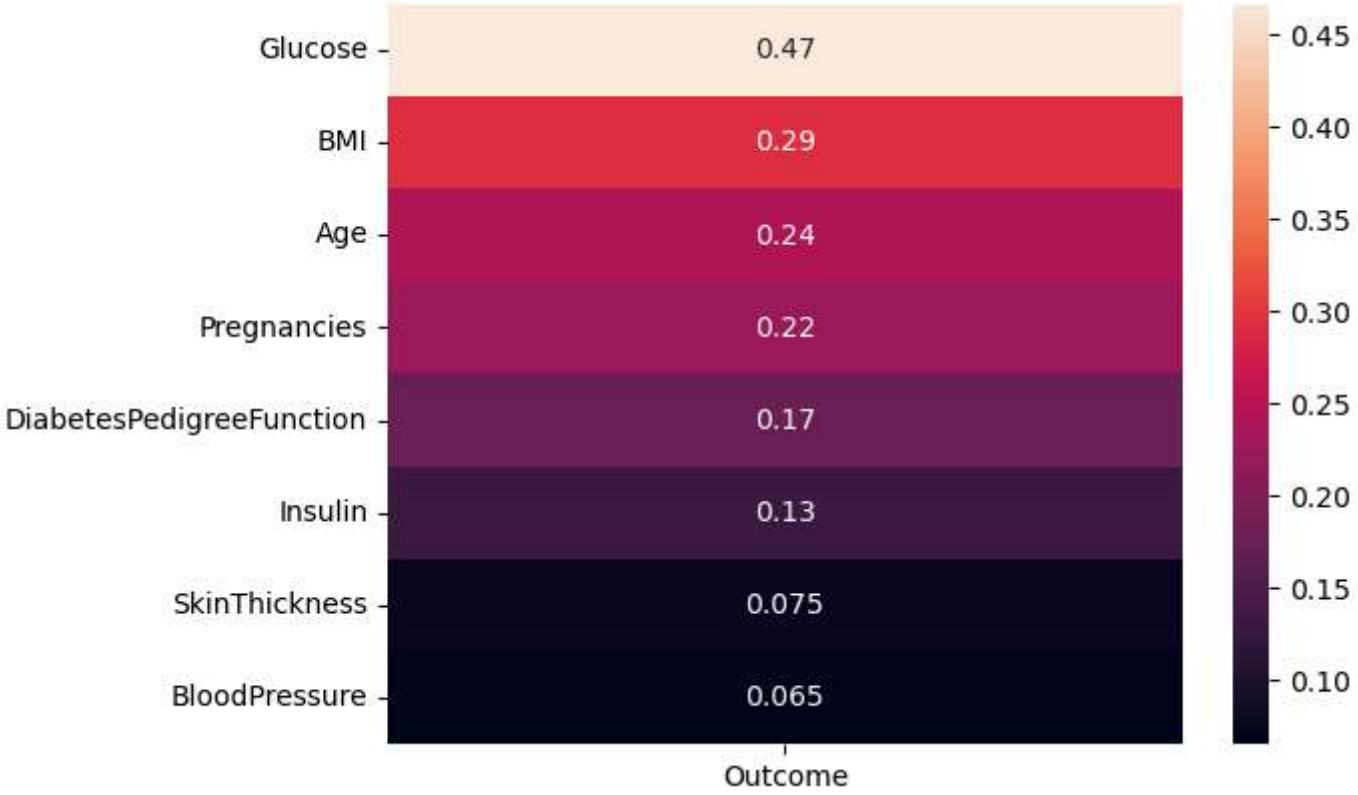

Pregnancies
Glucose
BloodPressure
SkinThickness
Insulin
BMI
DiabetesPedigreeFunction
Age
Outcome



In [22]:
```
X = Diabetes.drop(columns='Outcome')
y = Diabetes['Outcome']
```

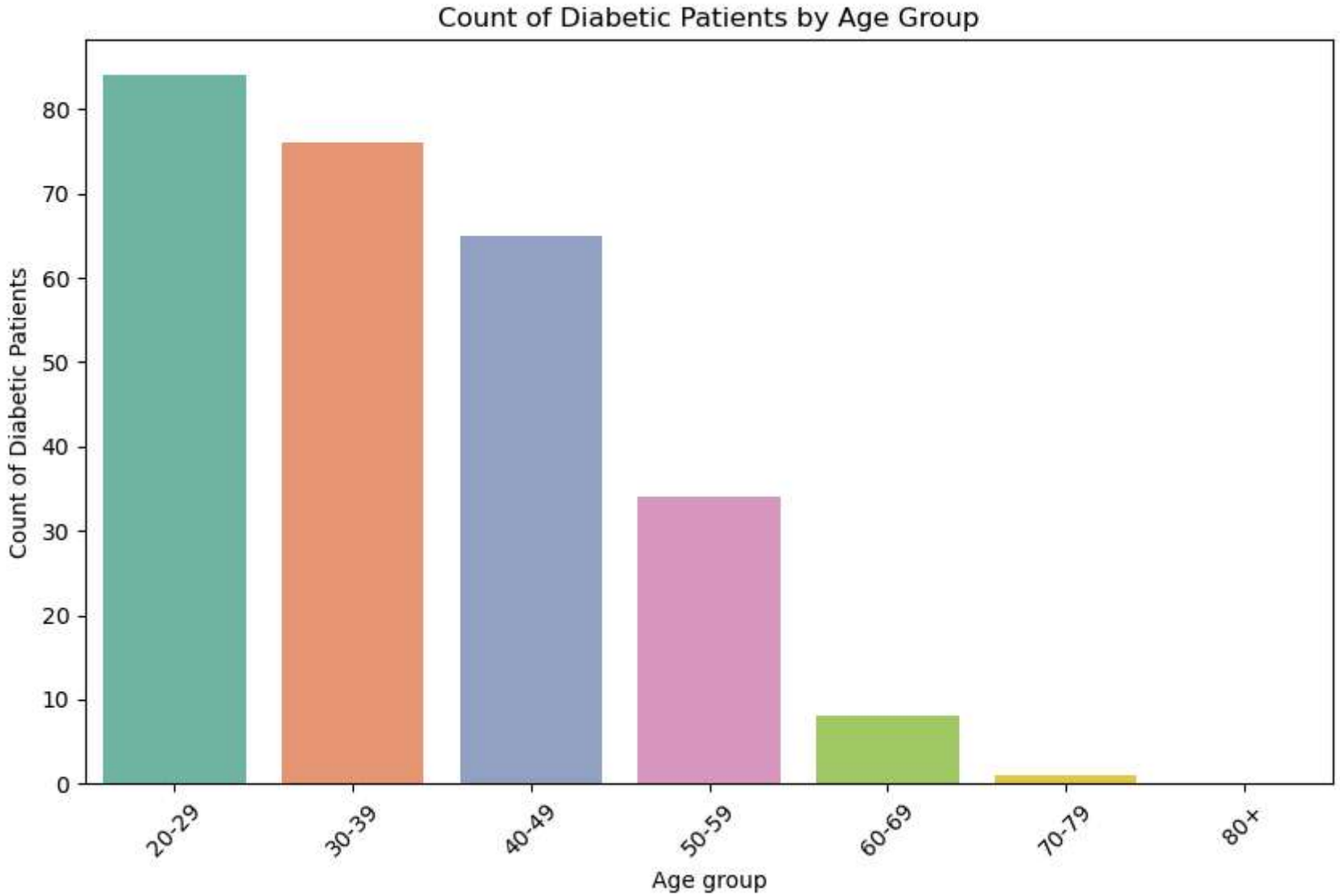
In [23]:
```
corr = Diabetes.corr()
target = corr['Outcome'].drop('Outcome')
sort_target = target.sort_values(ascending=False)
sns.heatmap(sort_target.to_frame(),annot=True)
```

Out[23]: <Axes: >



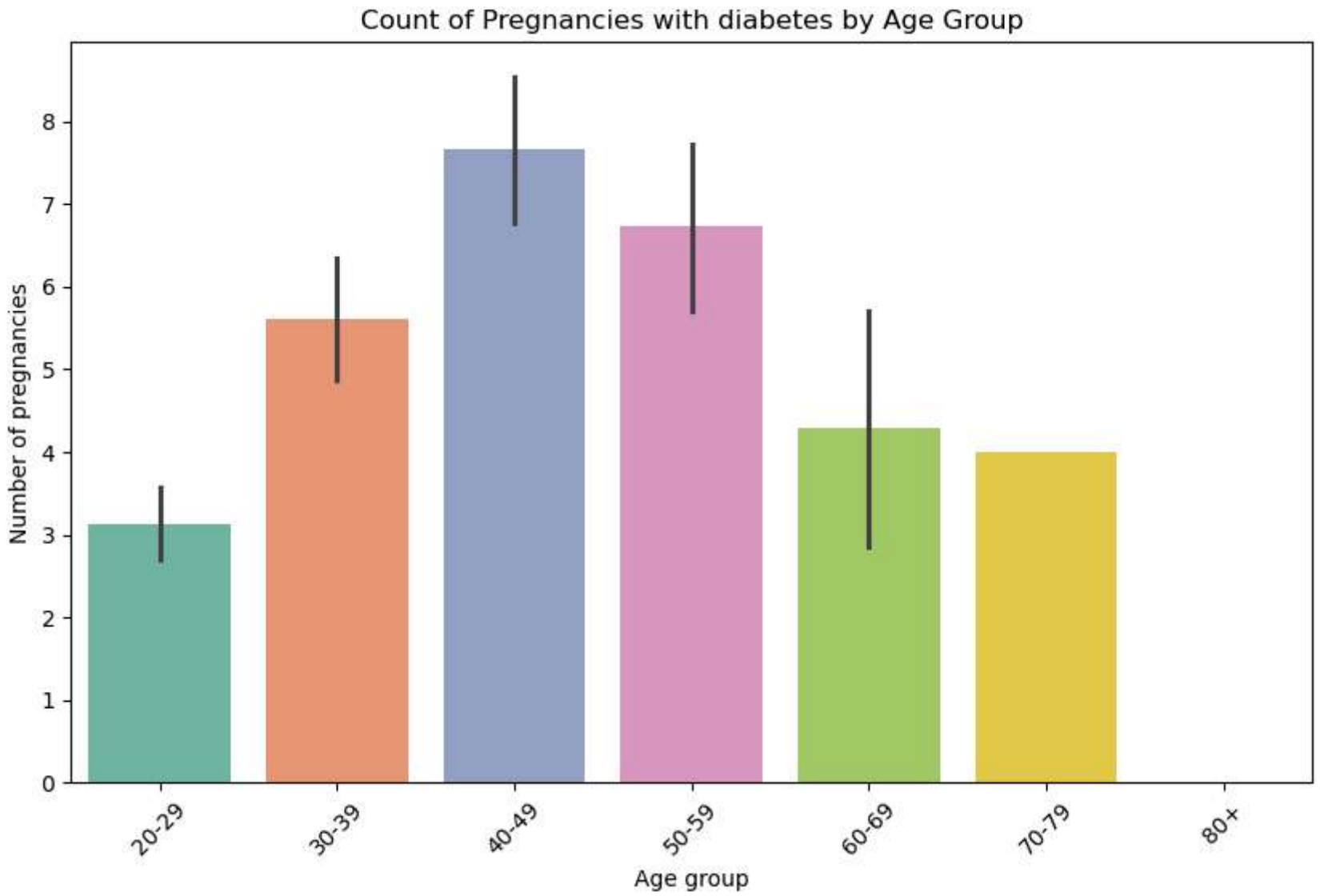
In [42]:
```
bins = [20, 30, 40, 50, 60, 70, 80, 200]
labels = ['20-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80+']
Diabetes['AgeGroup'] = pd.cut(Diabetes['Age'], bins=bins, labels=labels, right=False)
```

In [61]:
```
# Filter the dataset to include only records with 'Outcome' equal to 1 (Diabetic patients)
diabetic_df = Diabetes[Diabetes['Outcome'] == 1]

# Create a bar chart for Diabetic patients with age groups
plt.figure(figsize=(10, 6))
sns.countplot(
    data=diabetic_df,
    x='AgeGroup',
    order=labels,
    palette="Set2",
    hue='AgeGroup',
    dodge=False,      # ← important: avoids split bars
    legend=False      # ← optional: hides redundant legend
)
plt.xlabel('Age group')
plt.ylabel('Count of Diabetic Patients')
plt.title('Count of Diabetic Patients by Age Group')
plt.xticks(rotation=45)
plt.show()
```



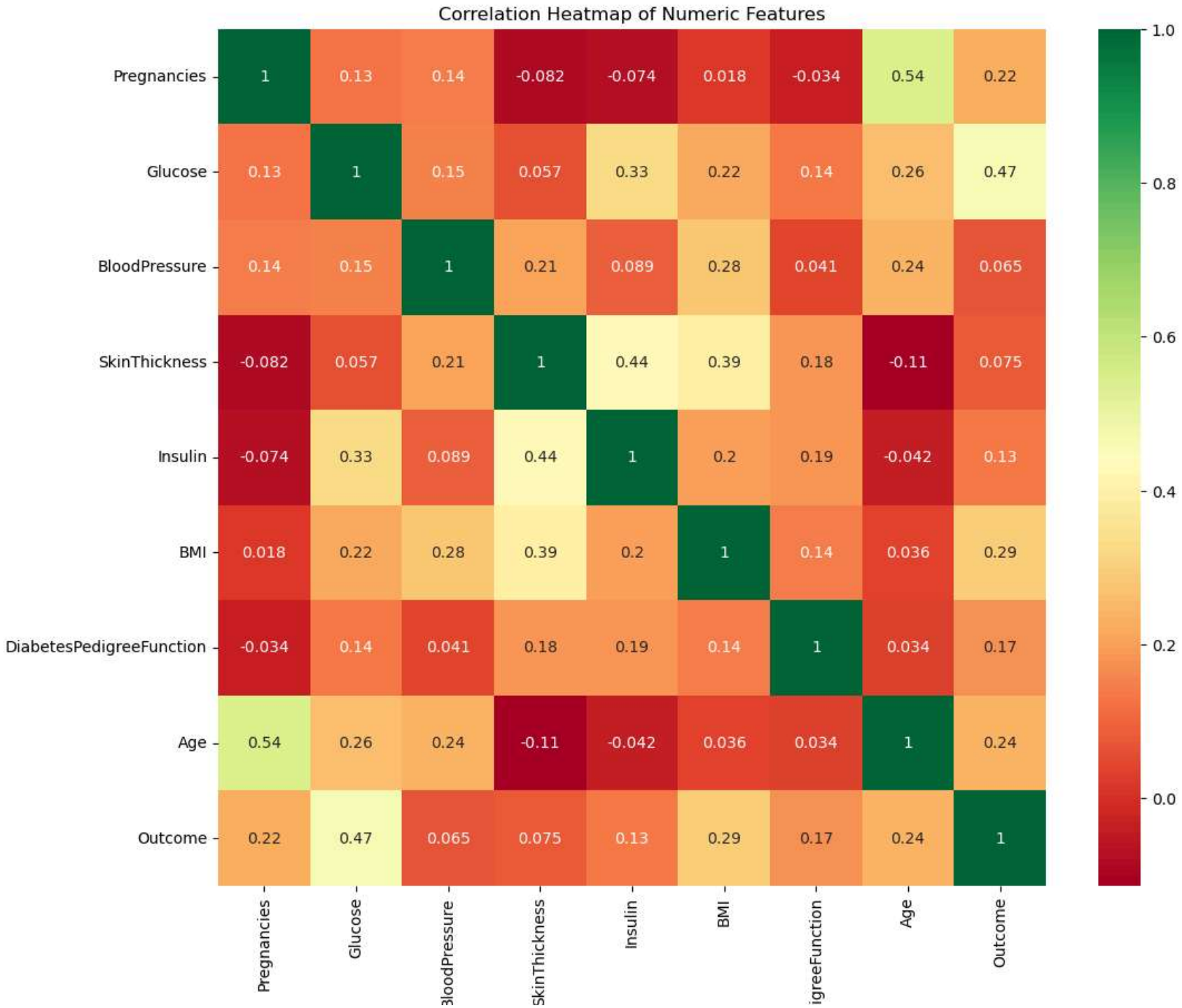
In [67]:
```
# Assuming 'AgeGroup' is a column in your original DataFrame 'Diabetes'
new_df = Diabetes[(Diabetes['Outcome'] == 1) & (Diabetes['Pregnancies'] > 0)]

# Create a bar chart with 'Outcome' as hue
plt.figure(figsize=(10, 6))
sns.barplot(data=new_df, x='AgeGroup', y='Pregnancies', hue='AgeGroup', palette='Set2', dodge=False, legend=False)
plt.xlabel('Age group')
```

```
plt.ylabel('Number of pregnancies')
plt.title('Count of Pregnancies with diabetes by Age Group')
plt.xticks(rotation=45)
plt.show()
```

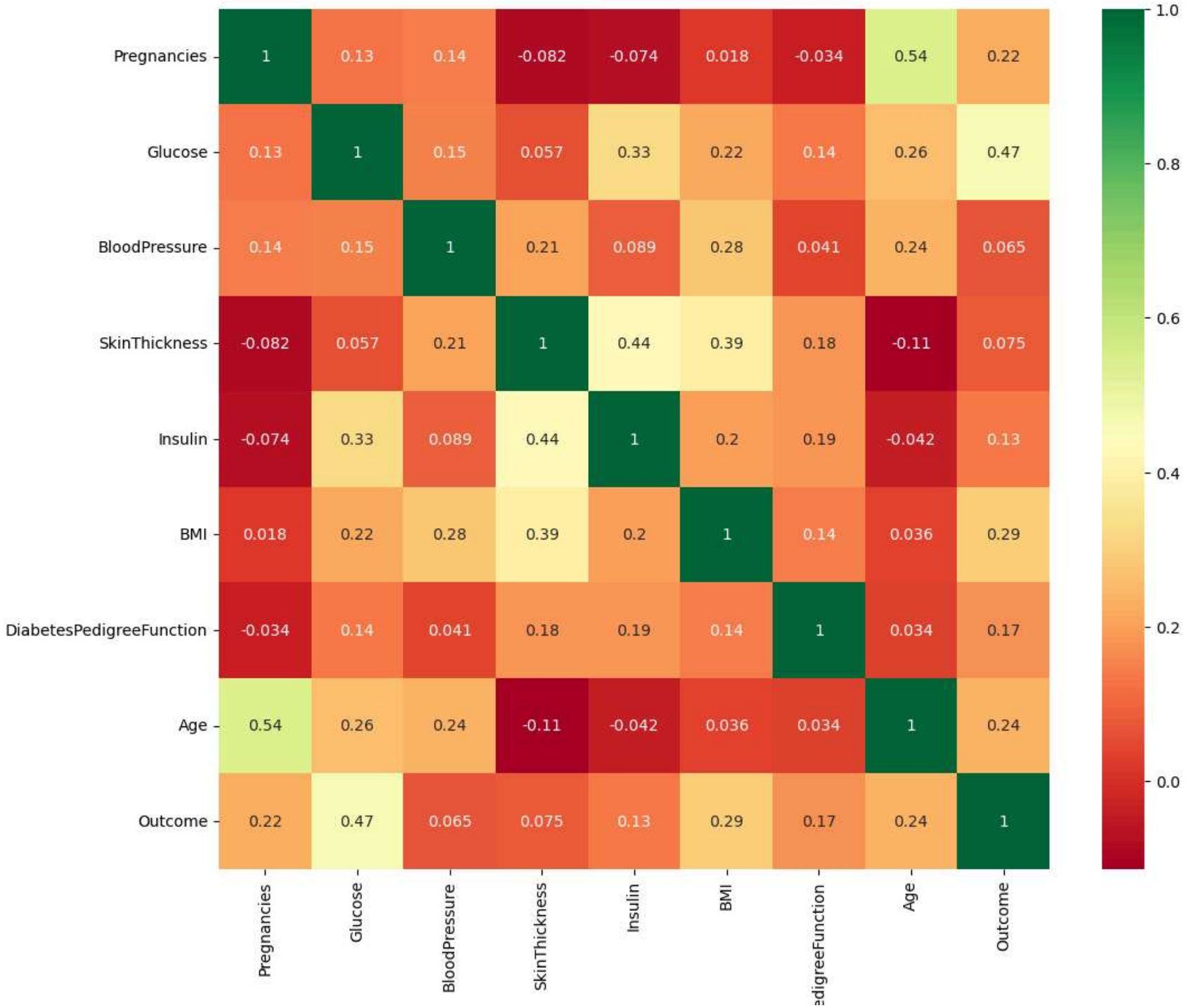


In [75]:
```
# Plotting the heatmap of correlation between all the feature before the cleaning
plt.figure(figsize=(12, 10))
numeric_df = Diabetes.select_dtypes(include='number')  # selects only numerical columns
```

```
sns.heatmap(numeric_df.corr(), annot=True, cmap="RdYlGn")
plt.title('Correlation Heatmap of Numeric Features')
plt.show()
```

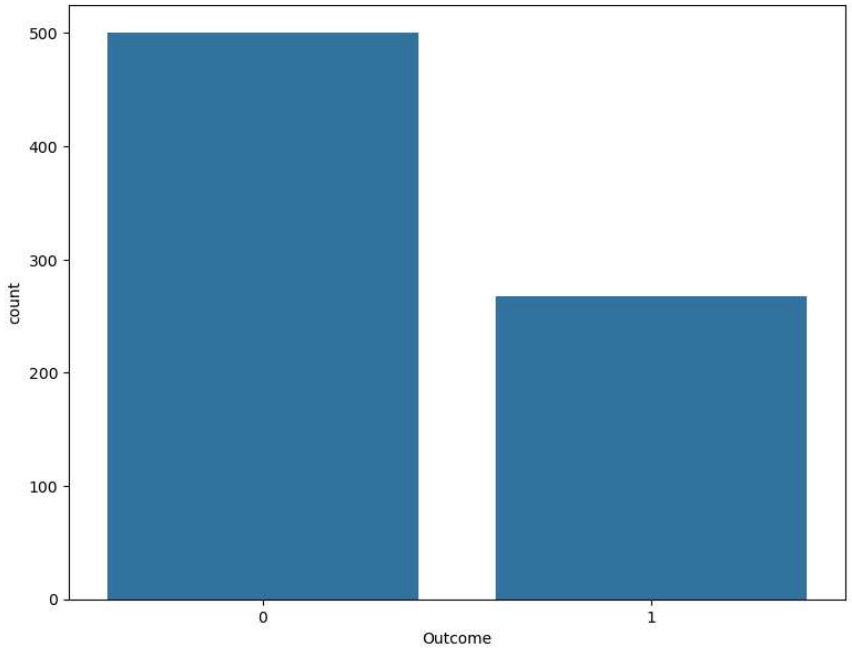

Correlation Heatmap of Numeric Features
Pregnancies 1 0.13 0.14 -0.082 -0.074 0.018 -0.034 0.54 0.22
Glucose 0.13 1 0.15 0.057 0.33 0.22 0.14 0.26 0.47
BloodPressure 0.14 0.15 1 0.21 0.089 0.28 0.041 0.24 0.065
SkinThickness -0.082 0.057 0.21 1 0.44 0.39 0.18 -0.11 0.075
Insulin -0.074 0.33 0.089 0.44 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
3loodPressure
SkinThickness
Insulin
BMI
igreeFunction
Age
Outcome
1.0
0.8
0.6
0.4
0.2
0.0

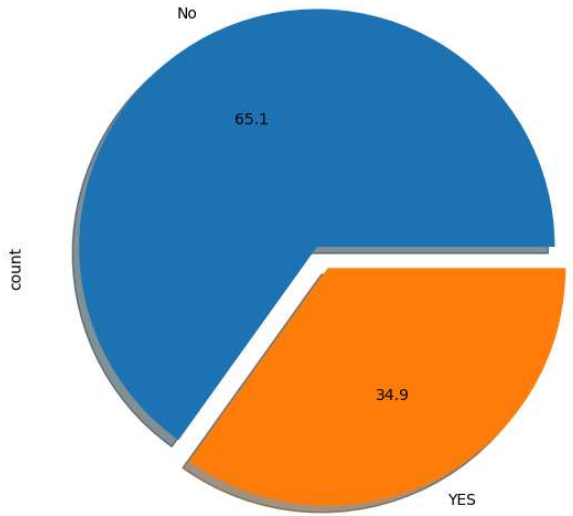
In [77]:
```
# Plotting the Heatmap of Correlation Between all the Feature After the Cleaning
plt.figure(figsize=(12,10))
P = sns.heatmap(Diabetes_copy.corr(), annot=True, cmap="RdYlGn")
plt.show()
```


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
edigreeFunction
Age
Outcome
1.0
0.8
0.6
0.4
0.2
0.0

DiabetesPe

In [79]:
```
fig,ax = plt.subplots(1,2, figsize=(20,7))
# Countplot
sns.countplot(data = Diabetes, x = "Outcome", ax = ax[0])
# Pie chart
Diabetes["Outcome"].value_counts().plot.pie(
    explode = [0.1,0],
    autopct = "%1.1f",
    labels = ["No", "YES"],
    shadow=True,
    ax = ax[1]
)
plt.show()
```





We observe from the above plot that:

- 65.1% patients in the dataset do NOT have diabetes.

- 34.9% patients in the dataset has diabetes.

Conclusion:-

1. It has a descent level of precision, indicating that when it predicts positive cases (diabetic). It's correct about 65% of the time.
2. Out of the 768 patients, 268 have been diagnosed with diabetes.
3. Patients with high blood pressure has greater chances of diabetes.
4. An increase in Blood pressure BMI and skin thickness also increases.
5. Increasing level of glucose and insulin increases chances of diabetes.