Diabetes prediction

In [1]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```python
df = pd.read_csv('diabetes.csv')
df.head()
```

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

In [3]:
```python
df.shape
```

Out[3]:
```
(768, 9)
```

In [4]:
```python
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 [5]:
```python
df.describe()
```

Out[5]:

	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 [6]:
```python
df.isna().sum()
```

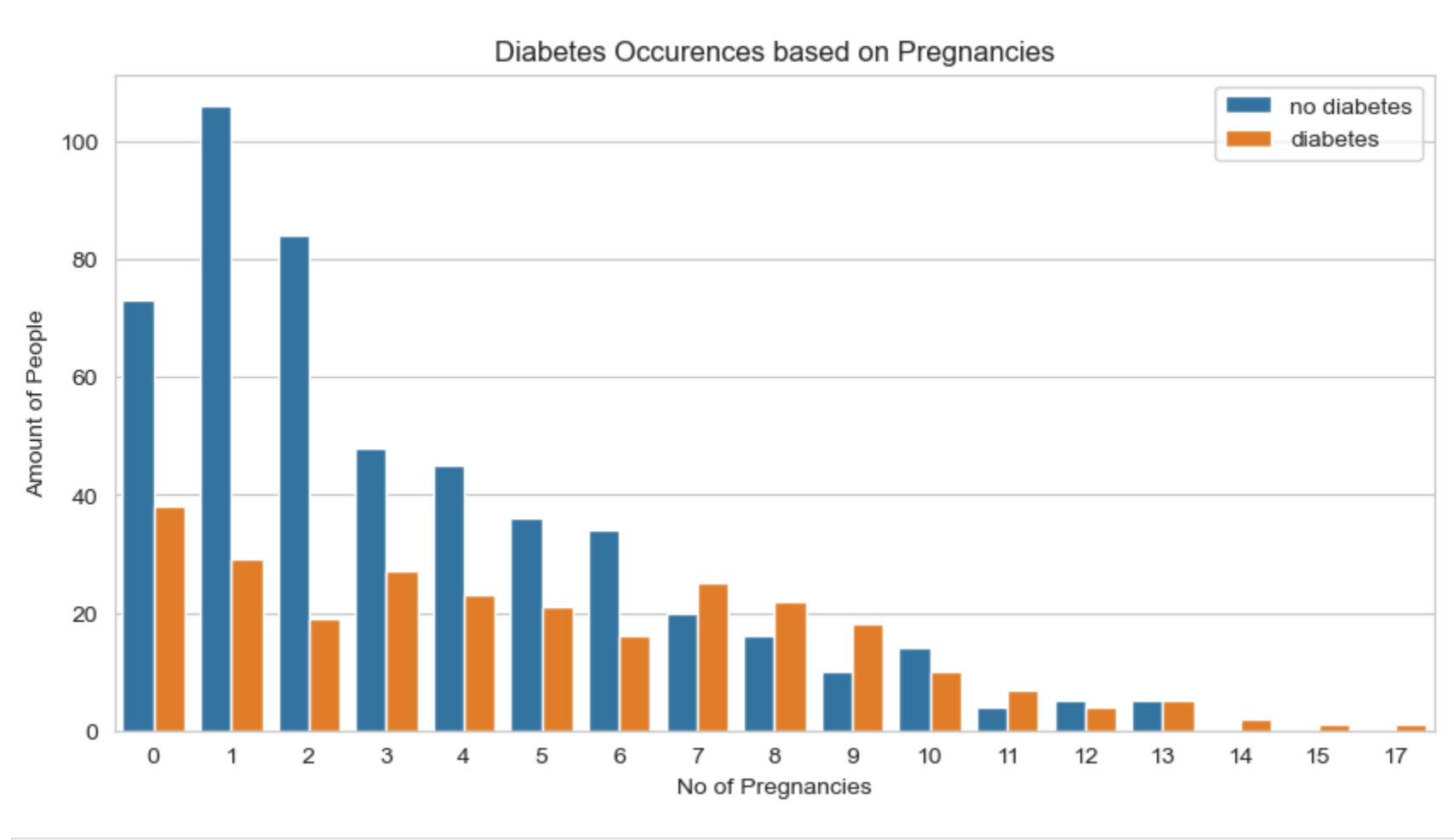
Out[6]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [7]:
```python
df['Pregnancies'].nunique()
```

Out[7]:
```
17
```

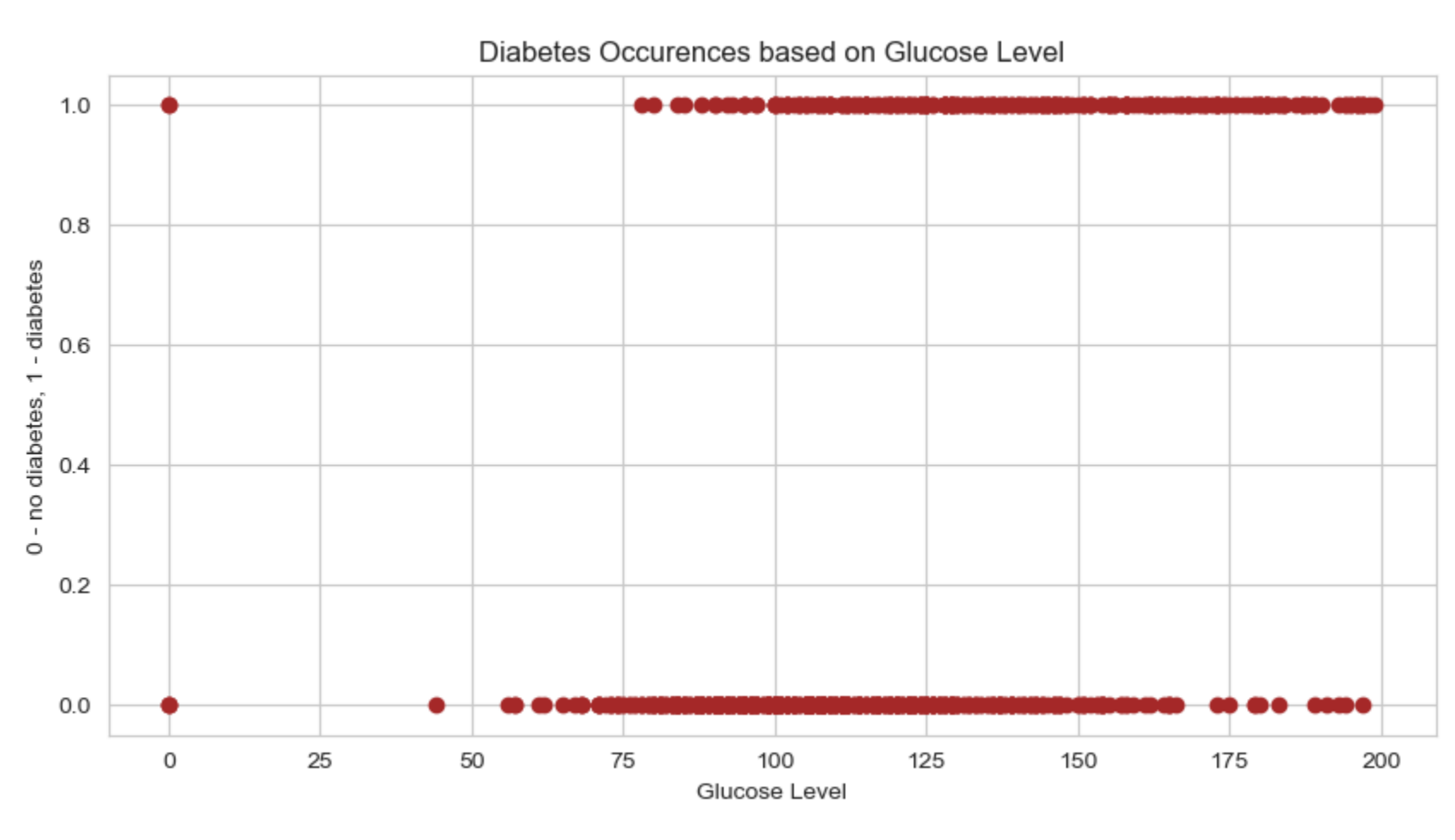
In [8]:
```python
plt.figure(figsize=(10, 5))
sns.set_style('whitegrid')
sns.countplot(data=df,
              x='Pregnancies',
              hue='Outcome')
plt.xlabel('No of Pregnancies')
plt.ylabel('Amount of People')
plt.legend(labels=['no diabetes', 'diabetes'])
plt.title('Diabetes Occurences based on Pregnancies')
```

Out[8]:
```
Text(0.5, 1.0, 'Diabetes Occurences based on Pregnancies')
```



In [9]:
```python
plt.figure(figsize=(10, 5))
plt.scatter(x=df['Glucose'],
            y=df['Outcome'],
            color='brown')
plt.xlabel('Glucose Level')
plt.ylabel('0 - no diabetes, 1 - diabetes')
plt.title('Diabetes Occurences based on Glucose Level')
```

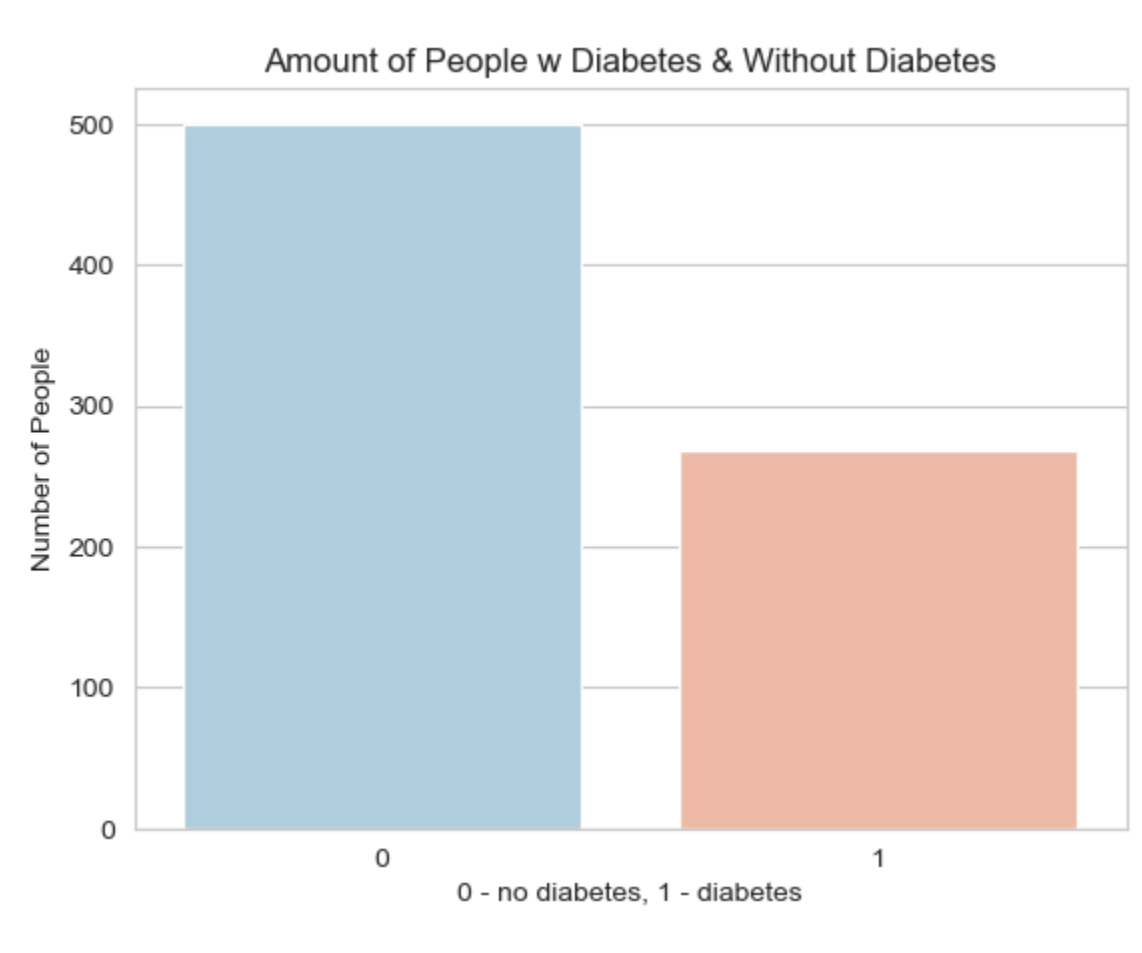
Out[9]:
```
Text(0.5, 1.0, 'Diabetes Occurences based on Glucose Level')
```



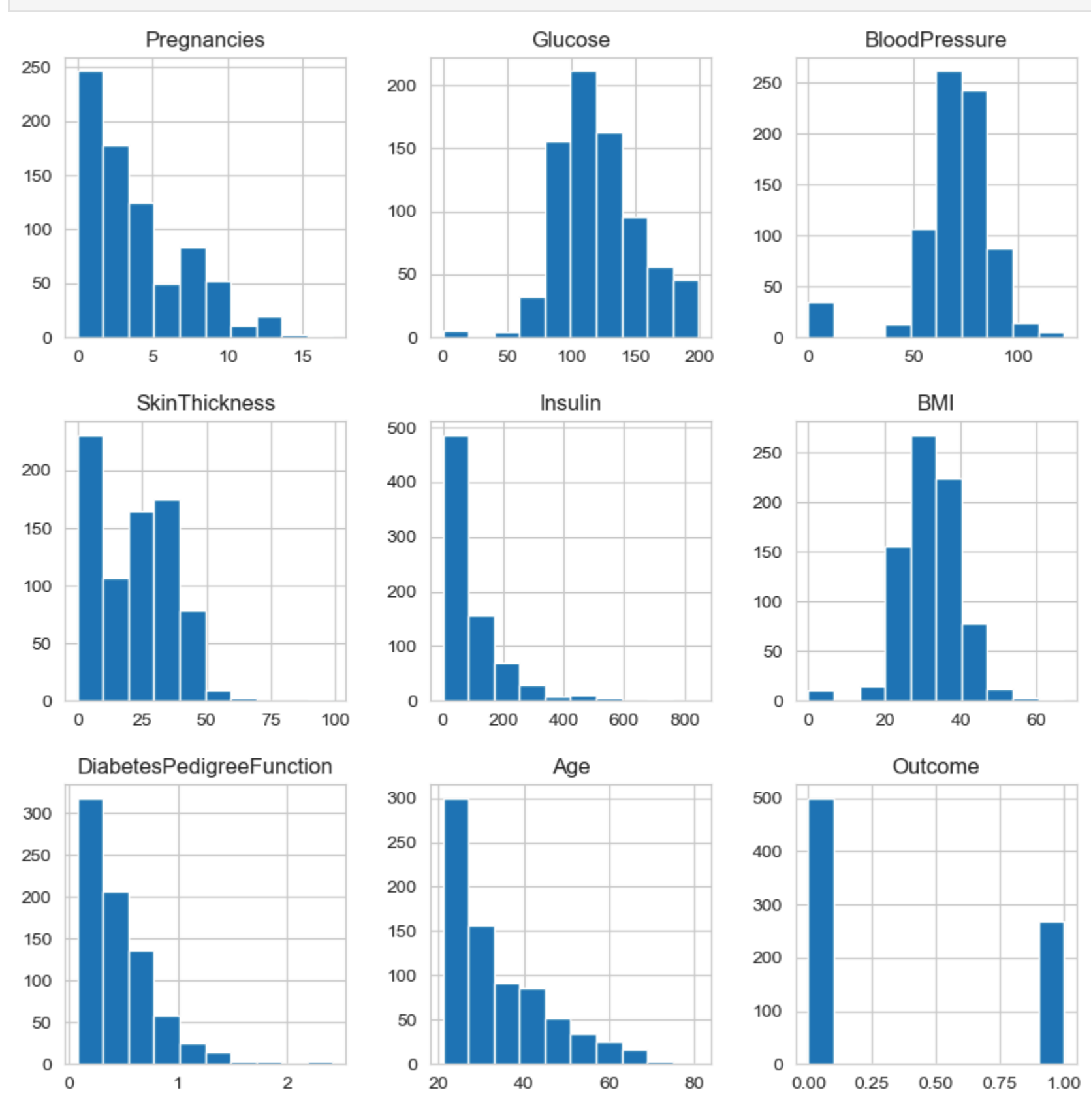
In [10]:
```python
sns.set_style('whitegrid')
sns.countplot(data=df,
              x='Outcome',
              palette='RdBu_r')
plt.title('Amount of People w Diabetes & Without Diabetes')
plt.xlabel('0 - no diabetes, 1 - diabetes')
plt.ylabel('Number of People')
```

```
C:\Users\ushaj\AppData\Local\Temp\ipykernel_9516\3445224849.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=df,
```

Out[10]:
```
Text(0, 0.5, 'Number of People')
```



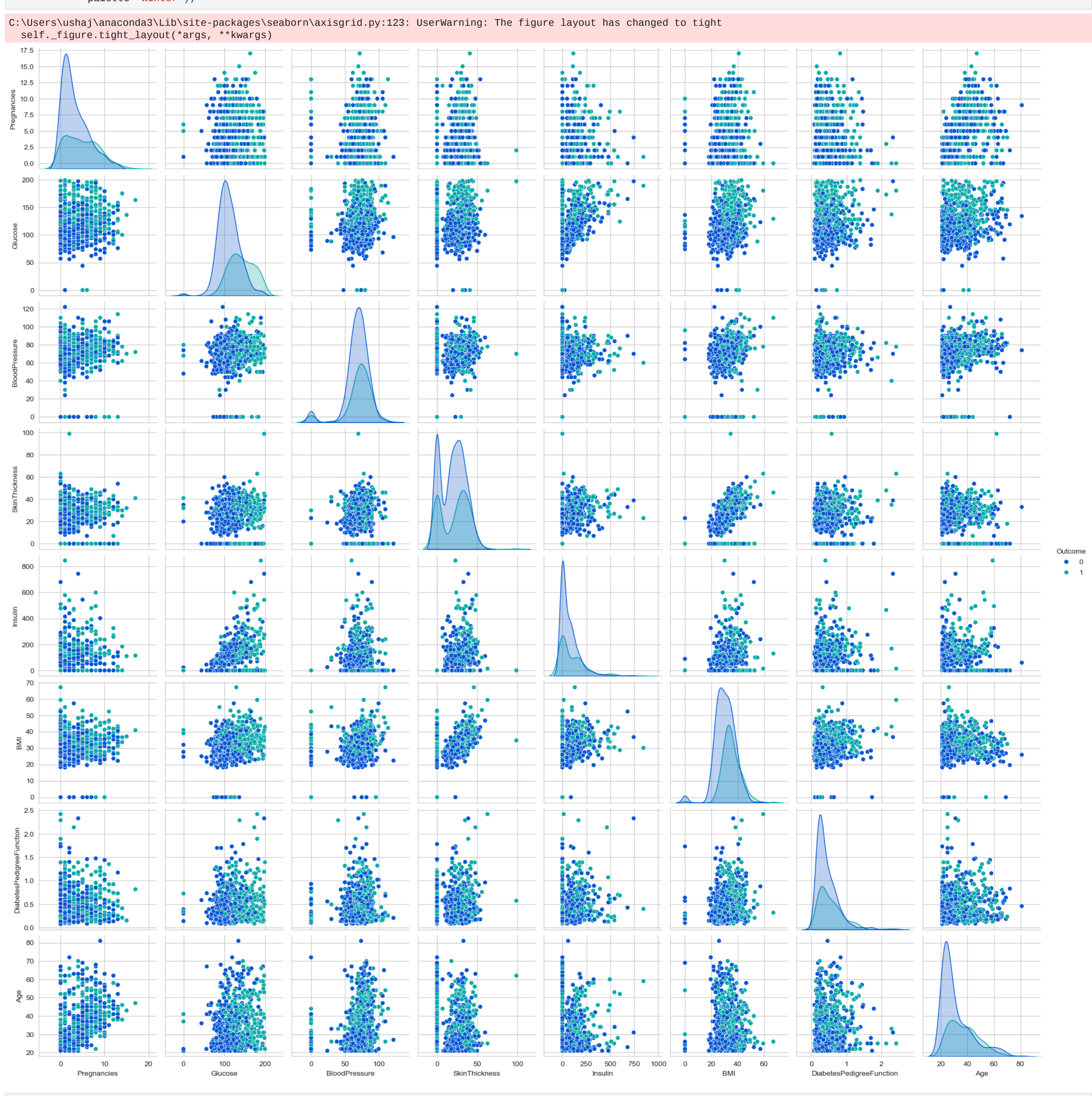
In [11]:
```python
df.hist(figsize=(10, 10));
```



In [12]:
```python
import warnings
warnings.filterwarnings("ignore", category=FutureWarning, module="seaborn._oldcore")
```

In [13]:
```python
sns.pairplot(data=df,
             hue='Outcome',
             palette='winter');
```

```
C:\Users\ushaj\anaconda3\Lib\site-packages\seaborn\axisgrid.py:123: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [14]:
```python
df.head(10)
```

Out[14]:

	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

In [15]:
```python
df1 = df.copy()
```

In [16]:
```python
df1[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df1[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']].replace(0,np.NaN)
```

In [17]:
```python
df1.isna().sum()
```

Out[17]:
```
Pregnancies                   0
Glucose                       5
BloodPressure                35
SkinThickness               227
Insulin                     374
BMI                          11
DiabetesPedigreeFunction      0
Age                           0
Outcome                       0
dtype: int64
```

In [18]:
```python
df1.head()
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	NaN	33.6	0.627	50	1
1	1	85.0	66.0	29.0	NaN	26.6	0.351	31	0
2	8	183.0	64.0	NaN	NaN	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.0	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.0	43.1	2.288	33	1

In [19]:
```python
def fill_missing_with_group_mean(df, columns, group_col):
    for column in columns:
        df[column] = df.groupby(group_col)[column].transform(lambda x: x.fillna(x.mean()))
    return df
```

In [20]:
```python
fill_missing_with_group_mean(df1, ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI'], 'Age')
```

Out[20]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.0	105.500000	33.6	0.627	50	1
1	1	85.0	66.0	29.0	222.333333	26.6	0.351	31	0
2	8	183.0	64.0	29.1	95.500000	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168.000000	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101.0	76.0	48.0	180.000000	32.9	0.171	63	0
764	2	122.0	70.0	27.0	144.285714	36.8	0.340	27	0
765	5	121.0	72.0	23.0	112.000000	26.2	0.245	30	0
766	1	126.0	60.0	29.0	98.333333	30.1	0.349	47	1
767	1	93.0	70.0	31.0	160.178571	30.4	0.315	23	0

768 rows × 9 columns

In [21]:
```python
df1.isna().sum()
```

Out[21]:
```
Pregnancies                  0
Glucose                      0
BloodPressure                1
SkinThickness               12
Insulin                     20
BMI                          0
DiabetesPedigreeFunction     0
Age                          0
Outcome                      0
dtype: int64
```

In [22]:
```python
df1[df1['Insulin'].isna()]
```

Out[22]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
123	5	132.0	80.0	NaN	NaN	26.8	0.186	69	0
129	0	105.0	84.0	58.0	NaN	27.9	0.741	62	1
148	5	147.0	78.0	37.0	NaN	33.7	0.218	65	0
221	2	158.0	90.0	NaN	NaN	31.6	0.805	66	1
294	0	161.0	50.0	37.0	NaN	21.9	0.254	65	0
362	5	103.0	108.0	37.0	NaN	39.2	0.305	65	0
363	4	146.0	78.0	NaN	NaN	38.5	0.520	67	1
453	2	119.0	NaN	NaN	NaN	19.6	0.832	72	0
456	1	135.0	54.0	58.0	NaN	26.7	0.687	62	0
489	8	194.0	80.0	NaN	NaN	26.1	0.551	67	0
495	6	166.0	74.0	NaN	NaN	26.6	0.304	66	0
509	8	120.0	78.0	NaN	NaN	25.0	0.409	64	0
537	0	57.0	60.0	NaN	NaN	21.7	0.735	67	0
552	6	114.0	88.0	NaN	NaN	27.8	0.247	66	0
579	2	197.0	70.0	99.0	NaN	34.7	0.575	62	1
582	12	121.0	78.0	17.0	NaN	26.5	0.259	62	0
666	4	145.0	82.0	18.0	NaN	32.5	0.235	70	1
674	8	91.0	82.0	NaN	NaN	35.6	0.587	68	0
684	5	136.0	82.0	NaN	NaN	26.8	0.640	69	0
759	6	190.0	92.0	NaN	NaN	35.5	0.278	66	1

In [23]:
```python
median_insulin_by_bmi = df1.groupby('BMI')['Insulin'].median()

# Function to fill missing Insulin values with median of corresponding BMI group
def fill_insulin(row):
    if pd.isna(row['Insulin']):
        return median_insulin_by_bmi.loc[row['BMI']]
    else:
        return row['Insulin']

# Apply the function to fill missing values
df1['Insulin'] = df1.apply(fill_insulin, axis=1)
```

In [24]:
```python
df1.isna().sum()
```

Out[24]:
```
Pregnancies                  0
Glucose                      0
BloodPressure                1
SkinThickness               12
Insulin                      2
BMI                          0
DiabetesPedigreeFunction     0
Age                          0
Outcome                      0
dtype: int64
```

In [25]:
```python
df1[df1['SkinThickness'].isna()]
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
123	5	132.0	80.0	NaN	144.285714	26.8	0.186	69	0
221	2	158.0	90.0	NaN	130.612903	31.6	0.805	66	1
363	4	146.0	78.0	NaN	125.000000	38.5	0.520	67	1
453	2	119.0	NaN	NaN	182.000000	19.6	0.832	72	0
489	8	194.0	80.0	NaN	106.360465	26.1	0.551	67	0
495	6	166.0	74.0	NaN	96.000000	26.6	0.304	66	0
509	8	120.0	78.0	NaN	130.612903	25.0	0.409	64	0
537	0	57.0	60.0	NaN	NaN	21.7	0.735	67	0
552	6	114.0	88.0	NaN	82.500000	27.8	0.247	66	0
674	8	91.0	82.0	NaN	378.000000	35.6	0.587	68	0
684	5	136.0	82.0	NaN	144.285714	26.8	0.640	69	0
759	6	190.0	92.0	NaN	145.400000	35.5	0.278	66	1

In [26]:
```python
age_group_df = df1[(df1['Age'] >= 60) & (df1['Age'] <= 70)]

# Calculate the median for SkinThickness within this age group
skinthickness_median = age_group_df['SkinThickness'].median()

print(f"SkinThickness median for age 60-70: {skinthickness_median}")
```

```
SkinThickness median for age 60-70: 33.0
```

In [27]:
```python
# Fill missing values in the original DataFrame using the median value
df1.loc[(df['Age'] >= 60) & (df['Age'] <= 70), 'SkinThickness'] = df1.loc[(df['Age'] >= 60) & (df['Age'] <= 70), 'SkinThickness'].fillna(skinthickness_median)
```

In [28]:
```python
df1.isna().sum()
```

Out[28]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               1
SkinThickness               1
Insulin                     2
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [29]:
```python
df1.dropna(inplace=True)
```

In [30]:
```python
df1.isna().sum()
```

Out[30]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [31]:
```python
df1.shape
```

Out[31]:
```
(765, 9)
```

Building a Model

In [32]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, cross_val_score

X = df1.drop('Outcome', axis=1)
y = df1['Outcome']

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

In [33]:
```python
log_reg = LogisticRegression(max_iter=500)

log_reg.fit(X_train, y_train)

log_reg.score(X_test, y_test)
```

Out[33]:
```
0.8300653594771242
```

In [34]:
```python
rfc = RandomForestClassifier()

rfc.fit(X_train, y_train)

rfc.score(X_test, y_test)
```

Out[34]:
```
0.8169934640522876
```

Hyperparameter Tuning

In [35]:
```python
from sklearn.model_selection import GridSearchCV

param_grid = {
    'penalty': ['l1', 'l2'],
    'C': [0.001, 0.01, 0.1, 1.0],
    'solver': ['liblinear', 'saga'],
    'max_iter': [100, 300, 500, 1000],
    'class_weight': [None, 'balanced']
}

gs_log_reg = GridSearchCV(log_reg,
                          param_grid,
                          cv=5,
                          n_jobs=-1)

gs_log_reg.fit(X_train, y_train)
```

Out[35]:
```
▸ GridSearchCV
▸ estimator: LogisticRegression
  ▸ LogisticRegression
```

In [36]:
```python
gs_log_reg.best_params_
```

Out[36]:
```
{'C': 1.0,
 'class_weight': 'balanced',
 'max_iter': 100,
 'penalty': 'l1',
 'solver': 'liblinear'}
```

In [37]:
```python
gs_log_reg.score(X_test, y_test)
```

Out[37]:
```
0.7647058823529411
```

In [38]:
```python
cross_val_score(log_reg, X, y).mean()
```

Out[38]:
```
0.7673202614379084
```

Correlation Matrix

In [39]:
```python
corr_mat = df1.corr()
corr_mat
```

Out[39]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.127449	0.213949	0.101023	0.097537	0.020027	-0.030932	0.563867	0.220037
Glucose	0.127449	1.000000	0.224578	0.200760	0.383126	0.232126	0.140591	0.280811	0.495727
BloodPressure	0.213949	0.224578	1.000000	0.189766	0.125405	0.283137	0.003456	0.347842	0.167525
SkinThickness	0.101023	0.200760	0.189766	1.000000	0.122240	0.531091	0.092407	0.178610	0.219839
Insulin	0.097537	0.383126	0.125405	0.122240	1.000000	0.134148	0.070413	0.273657	0.211156
BMI	0.020027	0.232126	0.283137	0.531091	0.134148	1.000000	0.155928	0.043379	0.312078
DiabetesPedigreeFunction	-0.030932	0.140591	0.003456	0.092407	0.070413	0.155928	1.000000	0.024248	0.176711
Age	0.563867	0.280811	0.347842	0.178610	0.273657	0.043379	0.024248	1.000000	0.251017
Outcome	0.220037	0.495727	0.167525	0.219839	0.211156	0.312078	0.176711	0.251017	1.000000

In [40]:
```python
plt.figure(figsize=(10, 7))
sns.heatmap(data=corr_mat, annot=True, cmap='viridis');
```