Dataset 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)
- Dataset Link : https://www.kaggle.com/datasets/mathchi/diabetes-data-set/code

In [68]:
```
#Import Libraries
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
```

In [69]:
```
# Import Data
df = pd.read_csv('diabetes.csv')
```

In [70]:
```
df.head()
```

Out[70]:

	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 [71]:
```
df.shape
```

Out[71]: (768, 9)

In [72]:
```
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 [73]:
```
df.describe()
```

Out[73]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunctio
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.00000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.47187
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.33132
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.07800
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.24375
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.37250
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.62625
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.42000

In [74]:
```
df.columns
```

Out[74]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [75]:
```
df.dtypes
```

Out[75]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

Applying Some EDA

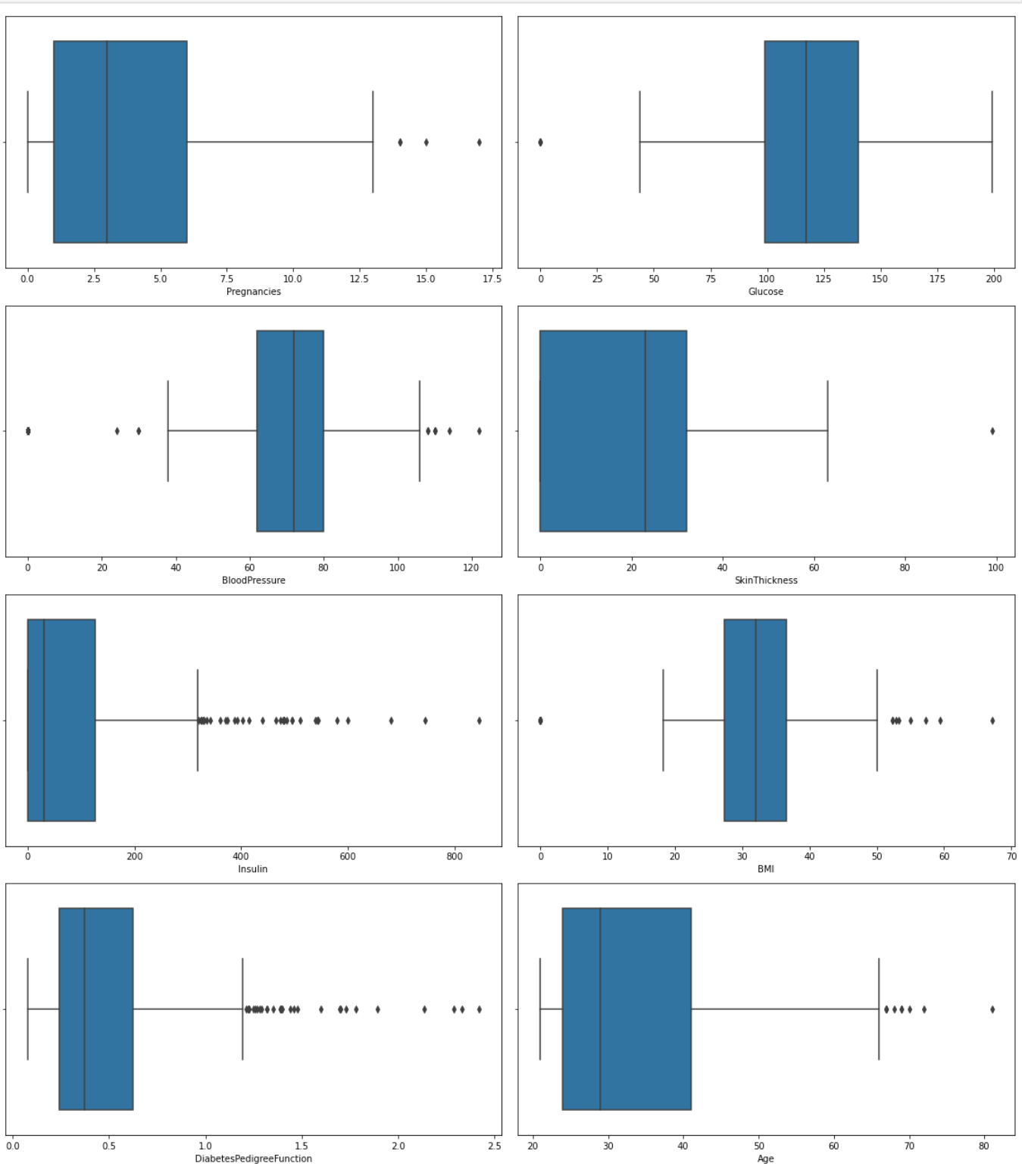
In [76]:
```
df.isnull().sum()
```

Out[76]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [77]:
```
fig, ax = plt.subplots(4, 2, figsize=(16,18))
axes_list = [axes for axes_row in ax for axes in axes_row]
column_list = list(df.columns)

for i, col in enumerate(column_list[:-1]):
    sns.boxplot(data=df, x=col, ax=axes_list[i])

plt.tight_layout()
plt.show()
```



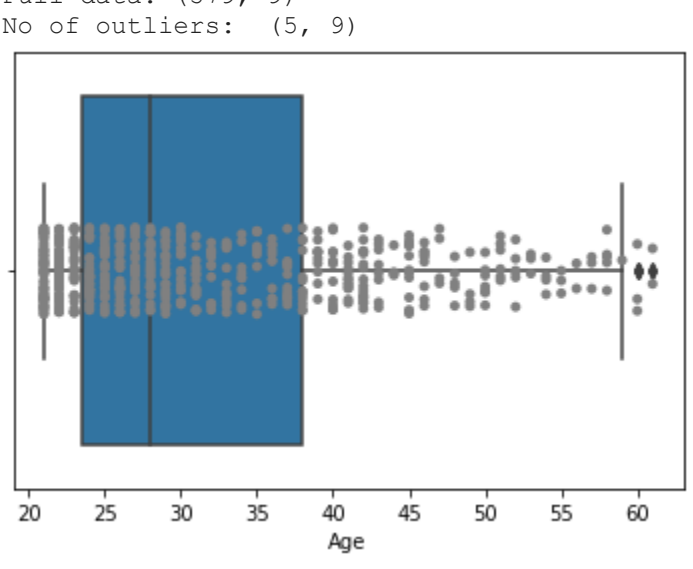
In [78]:
```
def check_outlier(data,col_name):
    """
    input:= data, column name
    output:= Lower wishker and Upper wishker
    """
    Q3 = data[col_name].quantile(0.75)
    Q1 = data[col_name].quantile(0.25)
    IQR = Q3-Q1
    print("75%:", Q3)
    print("25%",Q1)
    print("IQR:",IQR)

    LW = Q1 - 1.5*IQR
    UW = Q3 + 1.5*IQR
    print("Lower and Upper Wishker: ",LW, UW)
    print("Min and Max value: ", np.min(data[col_name]),np.max(data[col_name]))
    print("Full data:", data.shape)
    print("No of outliers: ",data[(data[col_name]<LW) | (data[col_name]>UW)].shape)

    sns.boxplot(x=data[col_name])
    sns.stripplot(x=data[col_name], color="0.5")
    plt.show()
    return LW, UW
```

In [66]:
```
# for c in df.columns:
# print("Column: ",c)
c = 'Age'
LW, UW= check_outlier(df,c)
# print("After removing outliers")
df=df[(df[c]>= LW) & (df[c]<= UW)]
# sns.boxplot(x=df[c])
# plt.show()
# print("="*40)
```

```
75%: 38.0
25% 23.5
IQR: 14.5
Lower and Upper Wishker:  1.75 59.75
Min and Max value:  21 61
Full data: (579, 9)
No of outliers:  (5, 9)
```



In [ ]: