1. INTRODUCTION

1.1 The dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases.

1.2 The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset.

In [1]:
```
import warnings
warnings.filterwarnings("ignore")

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
import matplotlib.pyplot as plt
import seaborn as sns
import imblearn
from imblearn.over_sampling import SMOTE
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc,f1_score
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.decomposition import PCA
```

2. DATA DESCRIPTION

In [2]:
```
path = r"diabetes.csv"
```

In [3]:
```
df = pd.read_csv(path)
df.head()
```

Out[3]:

	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

2.1(a) The datasets consists of several medical predictor variables (independent) and one target variable (dependent). Predictor variables includes the number of Pregnancies the patient has had, their BMI, their Glucose level, their BloodPressure, SkinThickness, Age, Insulin levels and Diabetes Pedigree Function while the target variable is Outcomes.

2.1(b) The two possible outcomes are : Patient is tested positive for diabetes -> Indicated by output “1” Patient is tested negative for diabetes -> Indicated by output “0”

In [4]:
```
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
```

2.2 This information and some research about the attributes gives us the insights about the different columns of the given dataset as :-

1. Pregnancies tells about the Number of times pregnant and is an integer variable.

1. Glucose is the Plasma glucose concentration (a 2 hours in an oral glucose tolerance test) and is an integer variable. STANDARD RANGE Hypoglycemia, also called low blood glucose or low blood sugar, occurs when the level of glucose in your blood drops below normal. For many people with diabetes, that means a level of 70 milligrams per deciliter (mg/dL) or less. Recommended range without diabetes is 70 to 130mg/dL. If blood glucose level is above 130mg/dL, that's Hyperglycemia.

1. BloodPressure is a Diastolic blood pressure (mm Hg) and is an integer variable. STANDARD RANGE 60-80 mm normal 80-89 mm Hypertension stage 90+ mm Hypertension crisis (case of diabetes)

1. SkinThickness tells about the Triceps skin fold thickness (mm) and is an integer variable. STANDARD RANGE Normal less than 30 mm 30+ Obese

1. Insulin is a 2-Hour serum insulin (mu U/ml) and is an integer variable. STANDARD RANGE Greater than 150 mu U/ml relates to insulin therapy

1. BMI is the Body mass index (weight in kg/(height in m)^2) and is an numerical variable. STANDARD RANGE Ideal Range between 18.5-24.9

1. DiabetesPedigreeFunction is an numerical variable.

1. Age given in years is an integer variable.

2.3 Data Summary

In [5]:
```
df.describe().T
```

Out[5]:

	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

2.3 Now since variables like Glucose, Blood Pressure, Skin Thickness, Insulin, BMI cannot be biologicaly 0 therefore, 0 is used as a missing value code, so replacing the 0 with NaN

We can fill missing code later in the code.

In [6]:
```
df['Glucose'] = np.where(df['Glucose'] == 0,np.NaN,df['Glucose'])
df['BloodPressure'] = np.where(df['BloodPressure'] == 0,np.NaN,df['BloodPressure'])
df['SkinThickness'] = np.where(df['SkinThickness'] == 0,np.NaN,df['SkinThickness'])
df['Insulin'] = np.where(df['Insulin'] == 0,np.NaN,df['Insulin'])
df['BMI'] = np.where(df['BMI'] == 0,np.NaN,df['BMI'])
```

So the summary is as follows after conversion:

In [7]:
```
df.describe().T
```

Out[7]:

	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	763.0	121.686763	30.535641	44.000	99.00000	117.0000	141.00000	199.00
BloodPressure	733.0	72.405184	12.382158	24.000	64.00000	72.0000	80.00000	122.00
SkinThickness	541.0	29.153420	10.476982	7.000	22.00000	29.0000	36.00000	99.00
Insulin	394.0	155.548223	118.775855	14.000	76.25000	125.0000	190.00000	846.00
BMI	757.0	32.457464	6.924988	18.200	27.50000	32.3000	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

SKEWENESS

In [8]:
```
# skewness along the index axis
df.skew(axis = 0, skipna = True)
```

Out[8]:
```
Pregnancies                 0.901674
Glucose                     0.530989
BloodPressure               0.134153
SkinThickness               0.690619
Insulin                     2.166464
BMI                         0.593970
DiabetesPedigreeFunction    1.919911
Age                         1.129597
Outcome                     0.635017
dtype: float64
```

KURTOSIS

In [9]:
```
# kurtosis along the index axis
df.kurt(axis = 0)
```

Out[9]:
```
Pregnancies                 0.159220
Glucose                    -0.277040
BloodPressure               0.911158
```

```
SkinThickness               2.935491
Insulin                     6.370522
BMI                         0.863379
DiabetesPedigreeFunction    5.594954
Age                         0.643159
Outcome                    -1.600930
dtype: float64
```

From the summary, skewness and kurtosis we can conclude that :-

1. Pregnancies
    - The number of Pregnancies vary from 0 to 17
    - IQR = 1 to 6 (covering 29% of the total range)
    - 25% of women have pregnancies > 6
    - Below 75% of the data i.e., distribution from minimum to 3rd Quartile covers 35% of the total range of the data. Hence, The data after 3rd Quartile is widely spread.
    - This shows that the data is moderately positively skewed with skewness = 0.8981549
    - The data is leptokurtic with kurtosis = 0.142184

1. Glucose
    - The Glucose concentration for women ranges from 44 to 199
    - IQR = 99 to 141 (covering 27% of the total range)
    - 25% 0f women have glucose concentration > 141
    - Below 75% of the data i.e., distribution from minimum to 3rd Quartile covers 63% of the total range of the data. Hence, The data after 3rd Quartile is lightly spread.
    - This shows that the data is approximately positively skewed with skewness = 0.5289026
    - The data is platykurtic with kurtosis = -0.290198
    - Hence, it is light tailed and have less outliers.
    - And there are 5 missing values.

1. BloodPressure
    - The Blood Pressure level ranges from 24 to 122
    - IQR = 64 to 80 (covering 16% of the total range)
    - 25% of women have blood pressure > 80
    - Below 75% of the data i.e., distribution from minimum to 3rd Quartile covers 57% of the total range of the data. Hence, The data after 3rd Quartile is moderately spread.
    - This shows that the data is approximately positively skewed with skewness = 0.1336042
    - The data is leptokurtic with kurtosis = 2.875579
    - Hence, it has contains more distribution in the tail and have more outliers
    - And there are 35 missing values.

1. SkinThickness
    - The Skin Thickness ranges from 7 to 99
    - IQR = 22 to 36 (covering 15% of the total range)
    - 25% of women have skin thickness > 36
    - Below 75% of the data i.e., distribution from minimum to 3rd Quartile covers 32% of the total range of the data. Hence, The data after 3rd Quartile is widely spread.
    - This shows that the data is approximately positively skewed with skewness = 0.6867941
    - The data is leptokurtic with kurtosis = 0.8861549
    - Hence, it has contains more distribution in the tail and have more outliers
    - And there are 227 missing values in this column.

1. Insulin
    - The Insulin level ranges from 14 to 846
    - IQR = 76.25 to 190 (covering 14% of the total range)
    - 25% of women have Insulin Level > 190
    - Below 75% of the data i.e., distribution from minimum to 3rd Quartile covers 21% of the total range of the data. Hence, The data after 3rd Quartile is widely spread.
    - This shows that the data is highly positively skewed with skewness = 2.149996
    - The data is leptokurtic with kurtosis = 6.227754
    - Hence, it has contains more distribution in the tail and have more outliers
    - And there are 374 missing values in this column.

1. BMI
    - The Body Mass Index ranges from 18.2 to 67.1
    - IQR = 27.5 to 36.6 (covering 19% of the total range)
    - 25% of women have Body Mass Index > 36.6
    - Below 75% of the data i.e., distribution from minimum to 3rd Quartile covers 38% of the total range of the data. Hence, The data after 3rd Quartile is moderately spread.
    - This shows that the data is moderately positively skewed with skewness = 0.5916179
    - The data is leptokurtic with kurtosis = 0.839607
    - Hence, it has contains more distribution in the tail and have more outliers
    - And there are 11 missing values in this column.

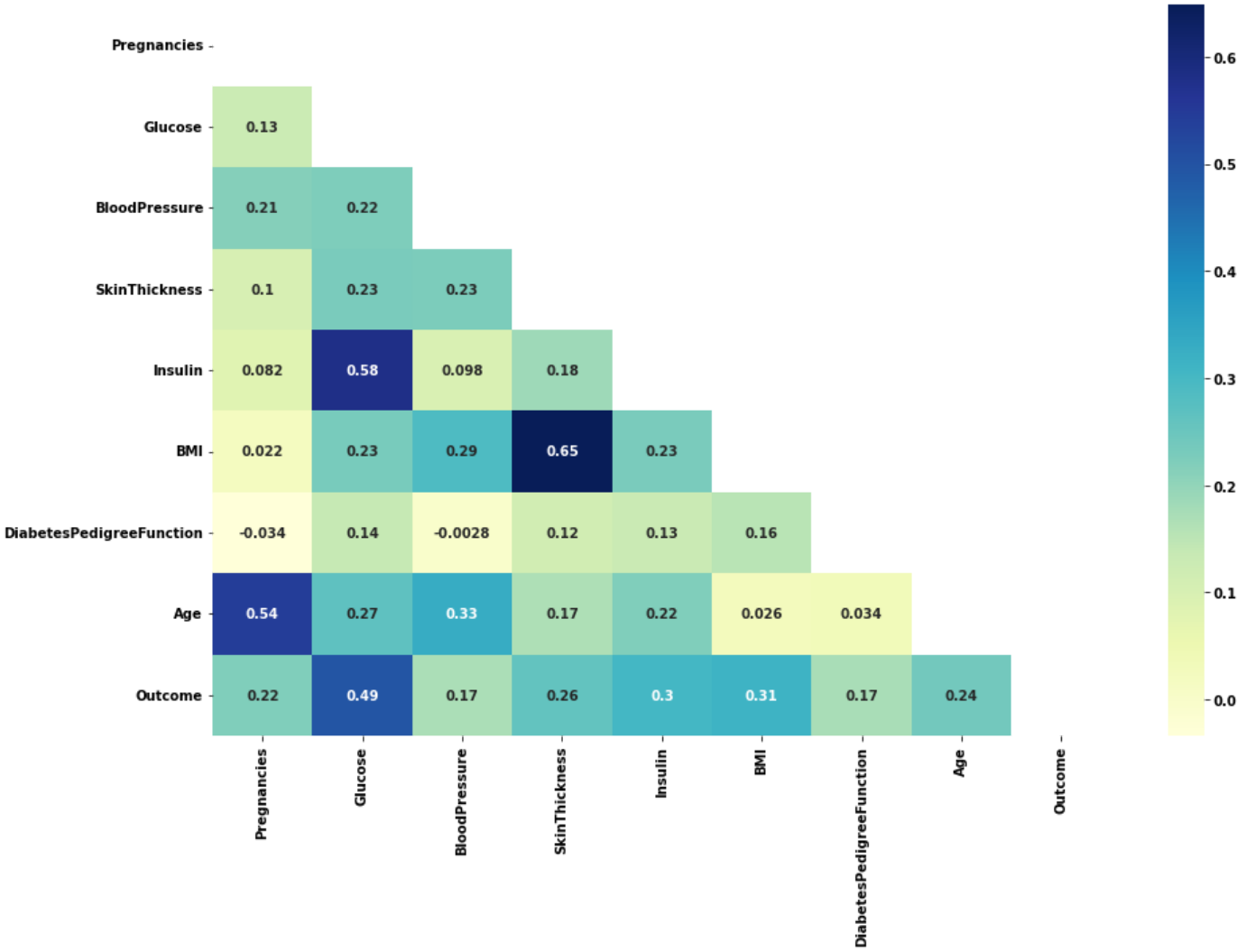
1. DiabetesPedigreeFunction
    - The Diabetes Pedigree Function from 0.78 to 2.42
    - IQR = 0.2437 to 0.6262 (covering 16% of the total range)
    - 25% of women have Diabetes Pedigree Function > 0.6262
    - Below 75% of the data i.e., distribution from minimum to 3rd Quartile covers 23% of the total range of the data. Hence, The data after 3rd Quartile is widely spread.
    - This shows that the data is highly positively skewed with skewness = 1.912418
    - The data is leptokurtic with kurtosis = 5.528539
    - Hence, it has contains more distribution in the tail and have more outliers

1. Age
    - The Age ranges from 21 to 81
    - IQR = 24 to 41 (covering 28% of the total range)
    - 25% of women have Age > 41
    - Below 75% of the data i.e., distribution from minimum to 3rd Quartile covers 33% of the total range of the data. Hence, The data after 3rd Quartile is spread.
    - This shows that the data is highly positively skewed with skewness = 1.125188
    - The data is leptokurtic with kurtosis = 0.6217269
    - Hence, it has contains more distribution in the tail and have more outliers

3. CORRELATION ANALYSIS

In [10]:
```
# import modules
plt.rcParams.update({'font.size': 18})
plt.rcParams["font.weight"] = "bold"
plt.figure(figsize=(15,10))
mask = np.triu(np.ones_like(df.corr()))
```

```
dataplot = sns.heatmap(df.corr(), cmap="YlGnBu", annot=True, mask=mask)
plt.show()
```



1. Pregnancy has low degree of correlation with other independent variables and shows a moderate degree of correlation with certain Age of the Females surveyed.

1. Glucose Concentration has a low degree of Correlation with all the other independent variables.

1. Blood Pressure has a low degree of Correlation with all the other independent variables.

1. Skin Thickness has a low degree of Correlation with all the other independent variables.

1. Insulin Level has a low degree of Correlation with all the other independent variables.

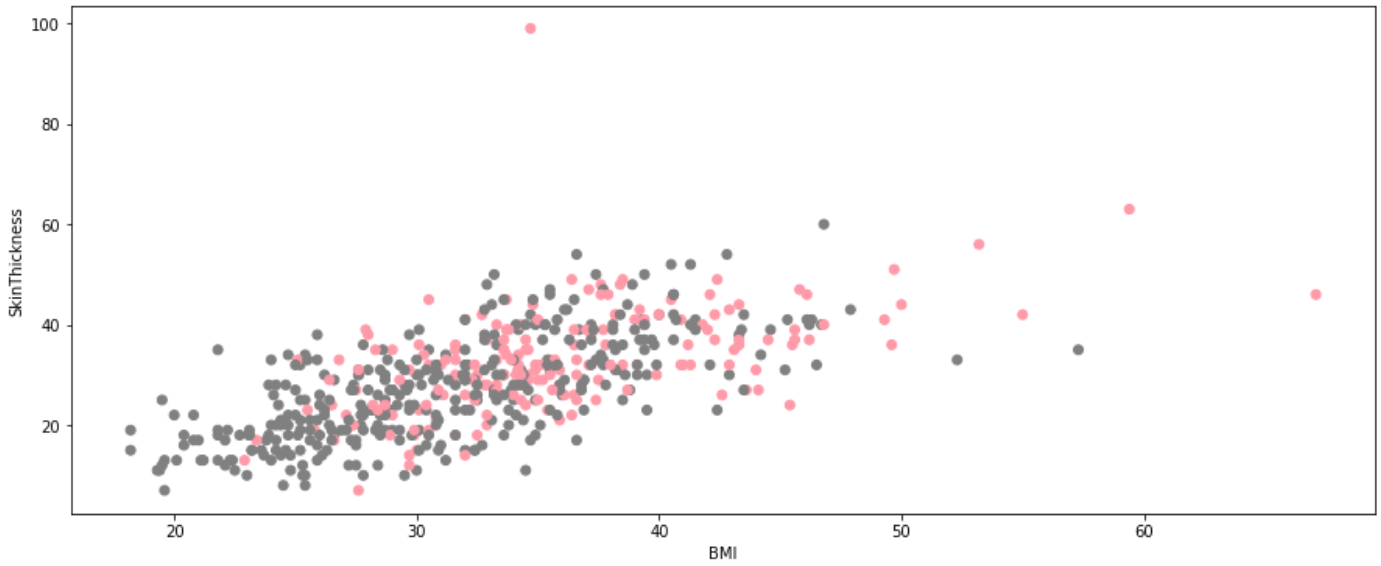
1. Diabetes Pedigree Function has a low degree of Correlation with all the other independent variables.

1. Outcome variable has high degree of correlation with Glucose Concentration and a moderate degree of correlation with Insulin and BMI.

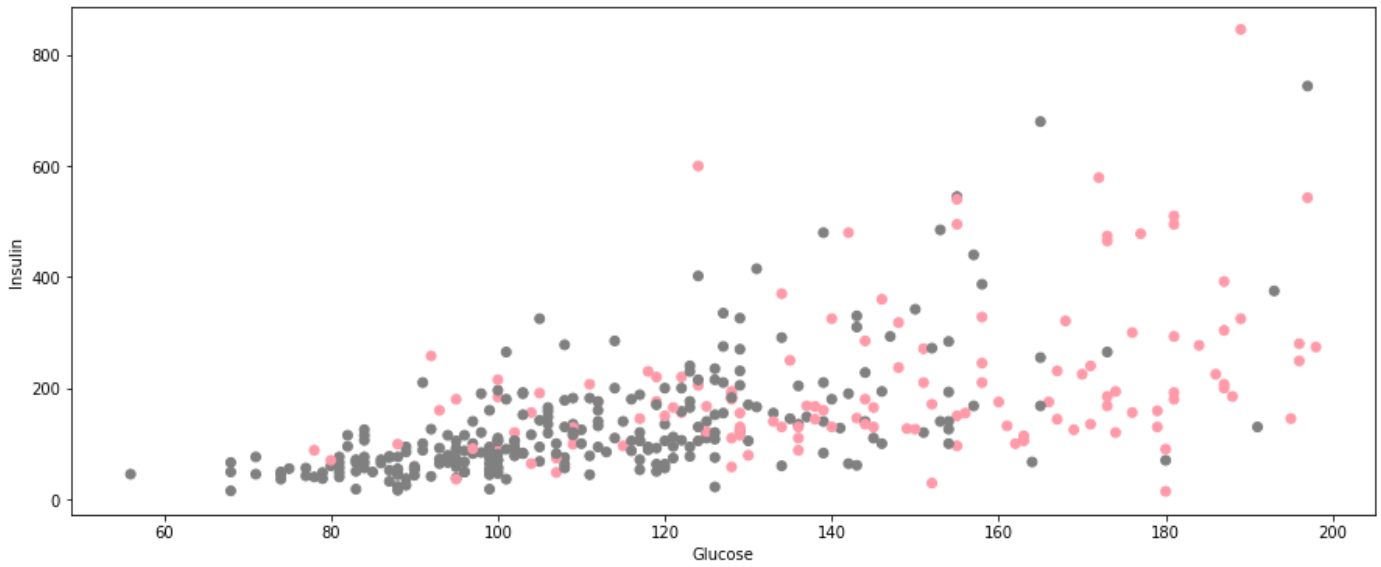
- BMI and Skin Thickness shows highest correlation of 0.65
- The second highest correlation can be seen between Insulin Level and Glucose Concentration
- Therefore, plotting a graph between them :

In [11]:
```
plt.rcParams.update({'font.size': 10})
```

```
plt.rcParams["font.weight"] = "normal"
plt.figure(figsize=(15,6))
colors = {0:'#808080', 1:'#ff9aa9'}
plt.scatter(df['BMI'], df['SkinThickness'], c=df['Outcome'].map(colors))
plt.xlabel("BMI")
plt.ylabel("SkinThickness")
plt.show()
```



In [12]:
```
plt.rcParams.update({'font.size': 10})
plt.rcParams["font.weight"] = "normal"
plt.figure(figsize=(15,6))
colors = {0:'#808080', 1:'#ff9aa9'}
plt.scatter(df['Glucose'], df['Insulin'], c=df['Outcome'].map(colors))
plt.xlabel("Glucose")
plt.ylabel("Insulin")
plt.show()
```



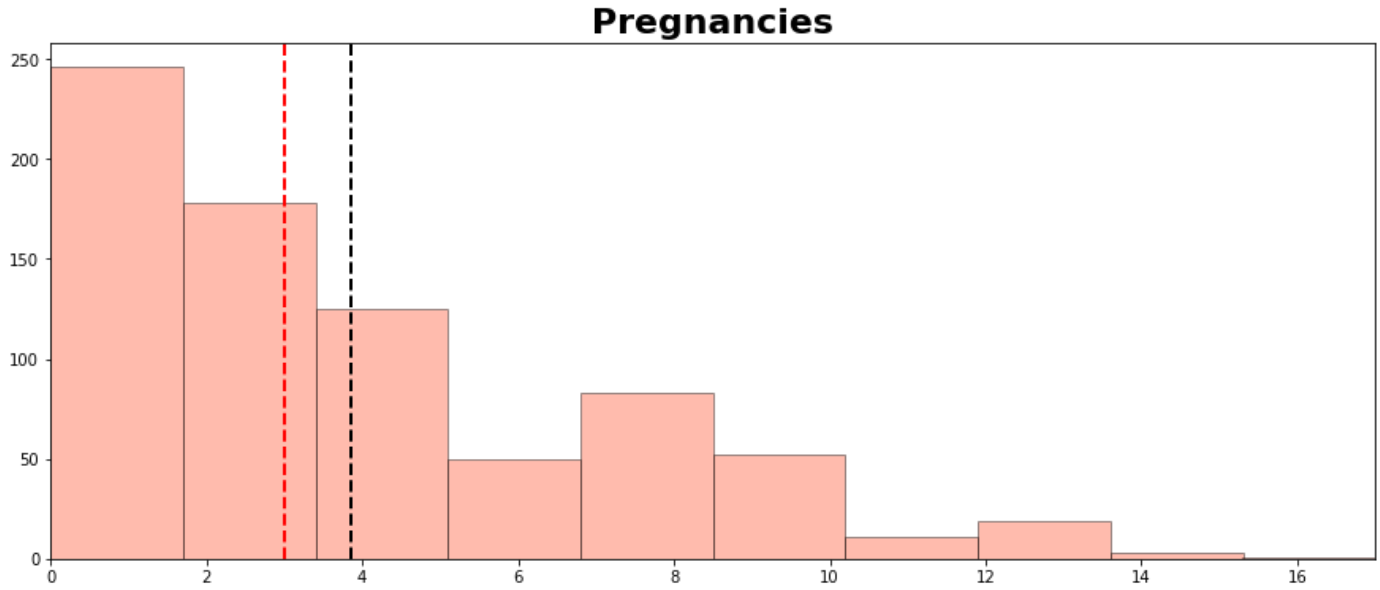
4. Univariate Analysis

BLACK LINE - Represents the mean value for an attribute

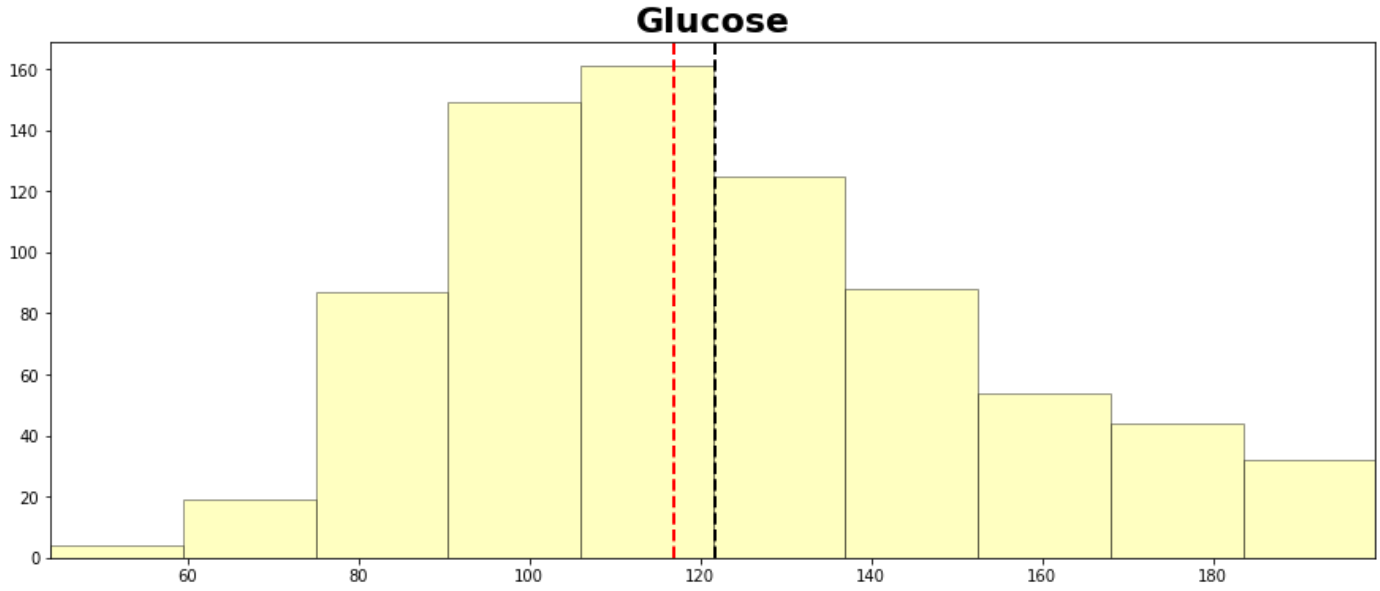
RED LINE - Represents the median value for an attribute

In [13]:
```
num_cols = len(df.columns)
```

```
colors = ['#FF5733', '#FFFF66', '#66FF66', '#66FFFF', '#6666FF', '#FF66FF', '#FFB6C1', '
for i in range(num_cols-1):
    x = df.iloc[:, i]
    name = x.name
    plt.figure(figsize=(15,6))
    result = plt.hist(x, color=colors[i], edgecolor='k', alpha=0.4)
    plt.axvline(x.mean(), color='k', linestyle='dashed', linewidth=2)
    plt.axvline(x.median(), color='red', linestyle='dashed', linewidth=2)
    plt.title("{}".format(name), fontweight='bold', fontsize=22)
    plt.xlim([x.min(), x.max()])
    plt.show();
    print('\n')
    Q1 = x.quantile(0.25)
    Q3 = x.quantile(0.75)
    IQR = Q3 - Q1
    threshold = 1.5 * IQR
    name = x.name
    outliers = df[(x < Q1 - threshold) | (x > Q3 + threshold)][name]
    print('OUTLIERS')
    print(outliers.values,'\n\n')
```

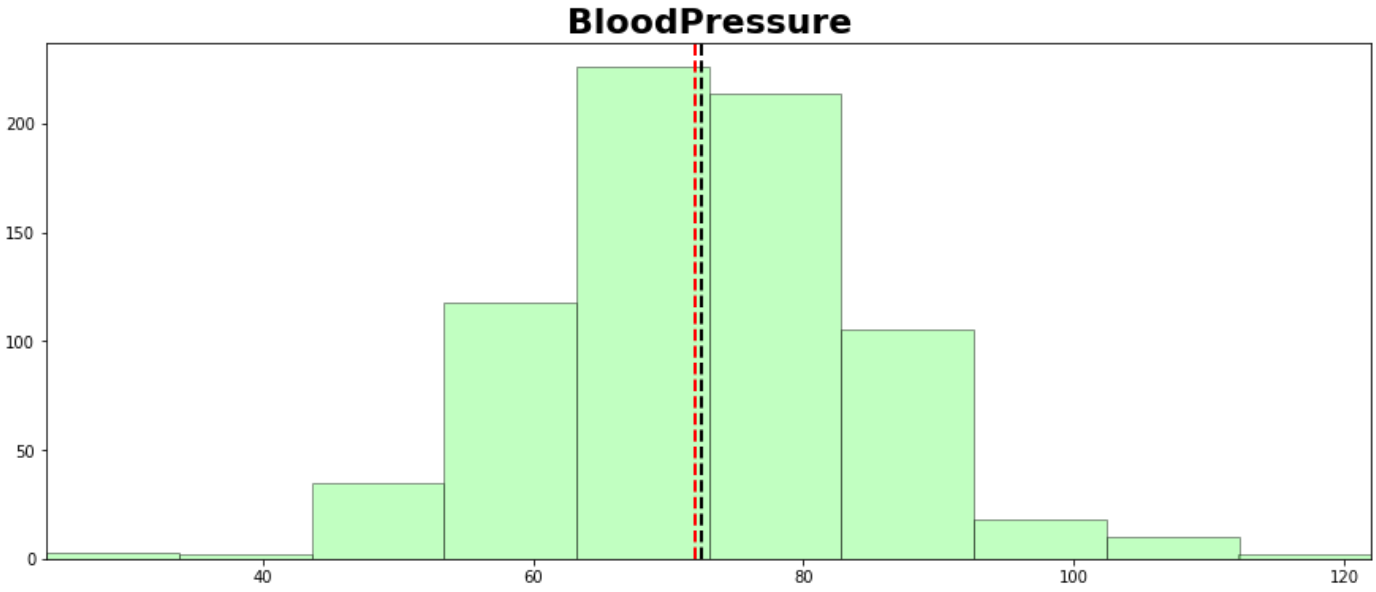



```
OUTLIERS
[15 17 14 14]
```

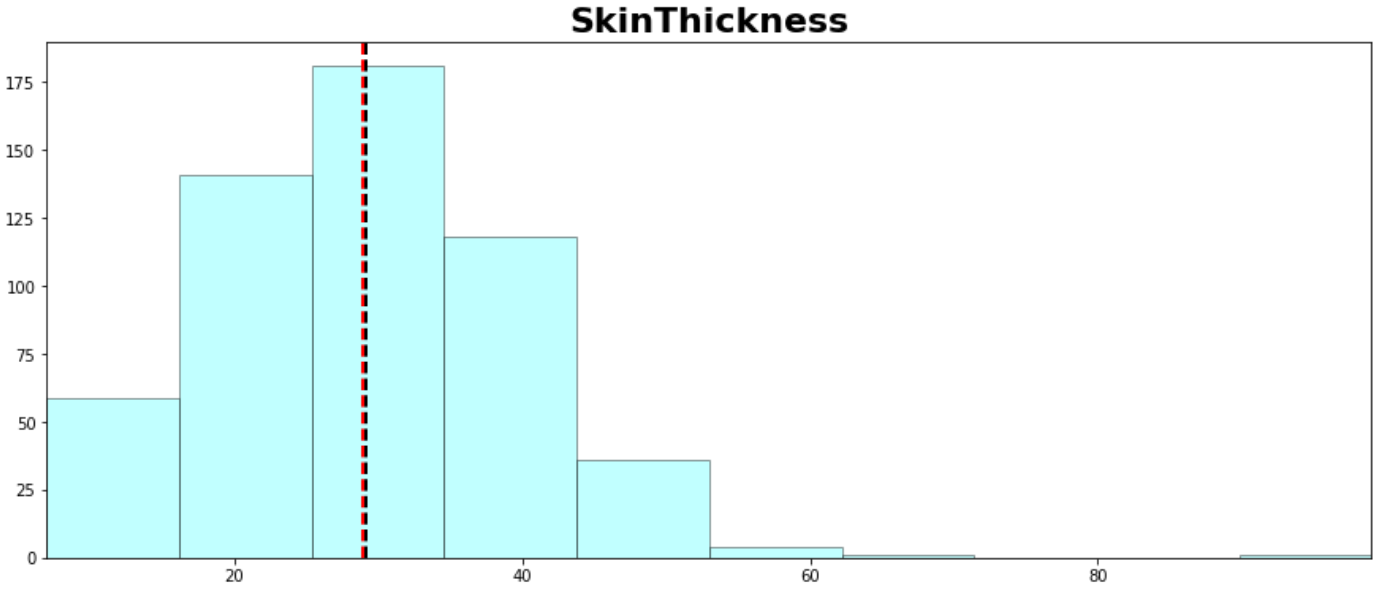



```
OUTLIERS
```

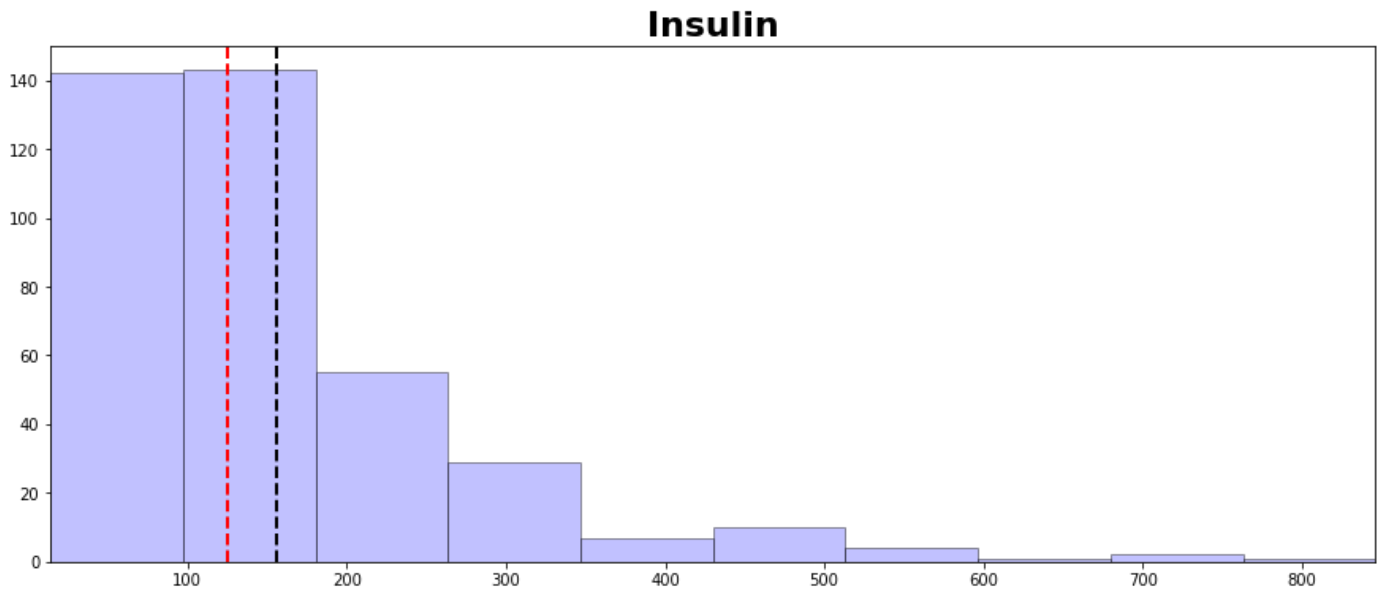
[]



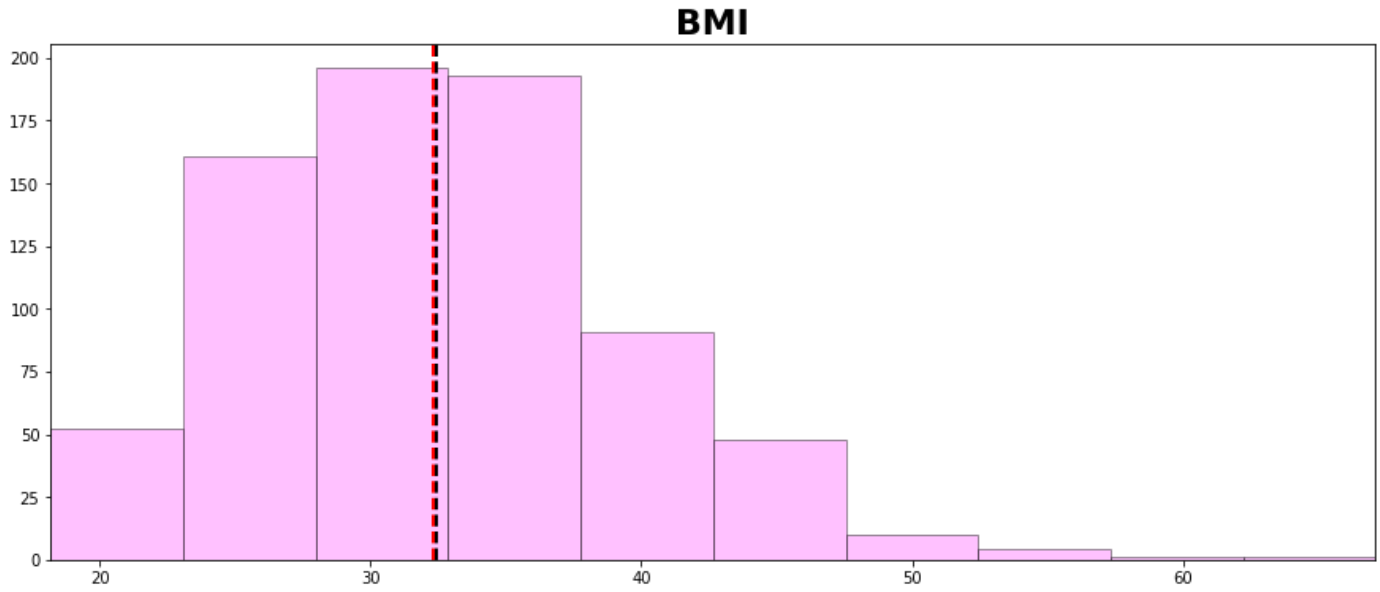
OUTLIERS
[30. 110. 108. 122. 30. 110. 108. 110. 24. 38. 106. 106. 106. 114.]



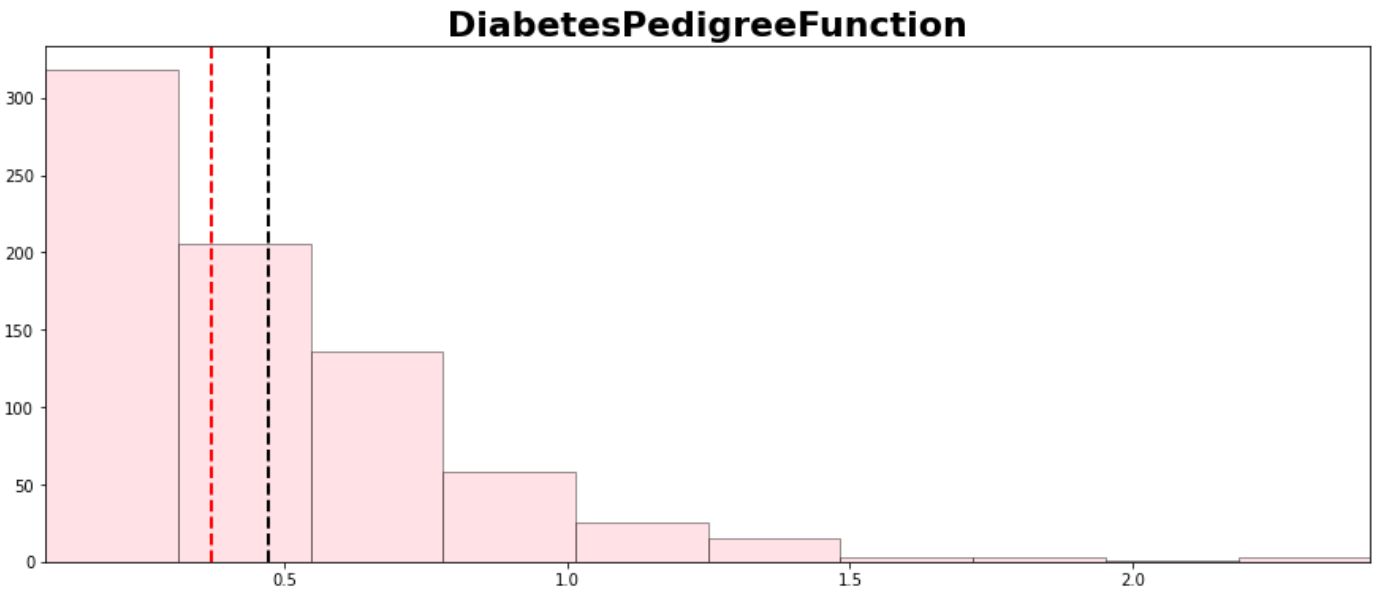
OUTLIERS
[60. 63. 99.]



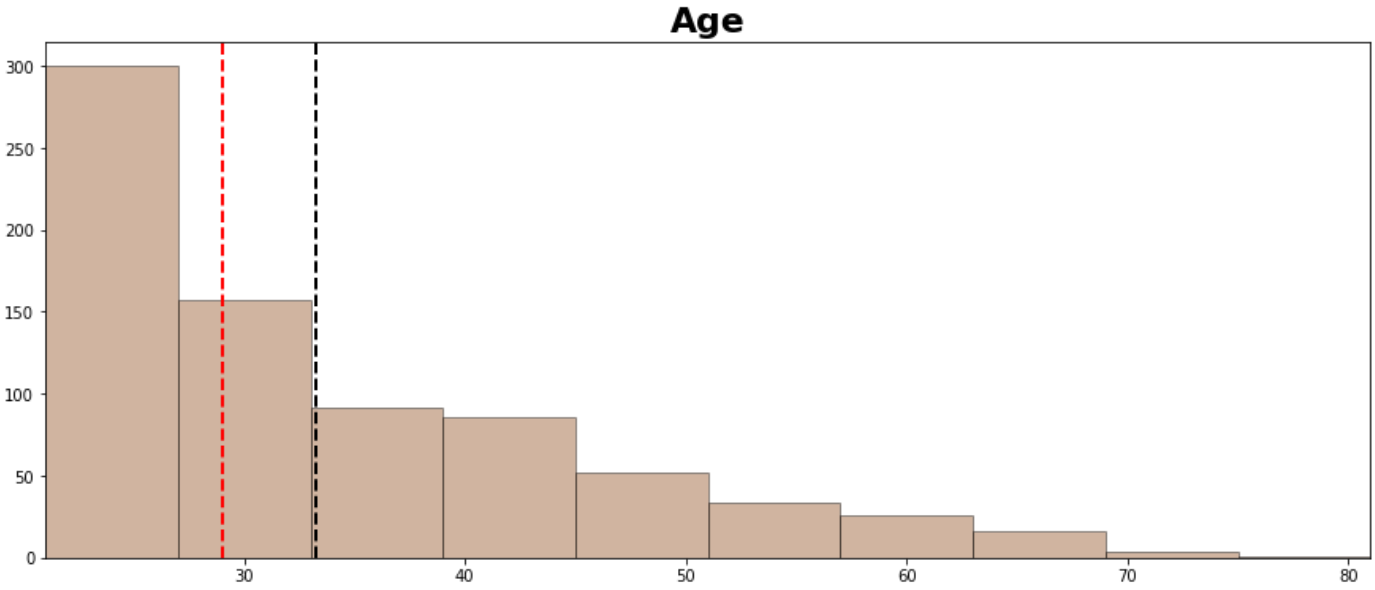
OUTLIERS
[543. 846. 495. 485. 495. 478. 744. 370. 680. 402. 375. 545. 465. 415.
 579. 474. 480. 600. 440. 540. 480. 387. 392. 510.]



OUTLIERS
[53.2 55. 67.1 52.3 52.3 52.9 59.4 57.3]




```
OUTLIERS
[2.288 1.441 1.39  1.893 1.781 1.222 1.4   1.321 1.224 2.329 1.318 1.213
 1.353 1.224 1.391 1.476 2.137 1.731 1.268 1.6   2.42  1.251 1.699 1.258
 1.282 1.698 1.461 1.292 1.394]
```




```
OUTLIERS
[69 67 72 81 67 67 70 68 69]
```

From the above graphs and outlier values we can conclude that:

1. Pregnancies:-
    - It ranges from 0 - 17
    - It is a moderately positively skewed distribution with skewness = 0.8981549
    - It is observed that as the number of pregnancies increases the frequency of women decreases. Since 5 is the threshold value for the number of pregnancies that covers 29% of the total range is more densely populated than the distribution > 5 pregnancies containing 219 women i.e., 28% of the total females surveyed.
    - The data is more widely dispersed after 10 pregnancies having 34 women.
    - 14th, 15th and 17th pregnancies are the outliers for this distribution.

1. Glucose Concentration :-
    - It ranges from 44 - 199
    - It is a approximately positively skewed distribution having skewness = 0.5289026
    - A large variation in frequencies can be seen for different values of Glucose Concentration hence, the data is widely dispersed about its range.
    - The distribution between 3rd quartile and the maximum value is more widely spread than the distribution between minimumn value and the 1st quartile.
    - It can be seen that maximum women (560 females) lie in the normal range i.e., between 70 to 140. Hence, this range of Glucose is densely populated.
    - There are 11 females having Glucose Concentration < 70 (which is the case of Hypoglycemia) and 192 females having Glucose Concentration > 140 (which is the case of Hyperglycemia)

1. Blood Pressure Level :-
    - It ranges from 24 - 122
    - It is a approximately positively skewed distribution having skewness = 0.1336042
    - A large variation in frequencies can be seen for different values of Blood Pressure Level hence, the data is widely dispersed about its range.
    - The distribution between 3rd quartile and the maximum value is more widely spread than the distribution between minimumn value and the 1st quartile.
    - It can be seen that maximum women (560 females) lie in the normal range i.e., between 70 to 140. Hence, this range of Glucose is densely populated.
    - There are 11 females having Glucose Concentration < 70 (which is the case of Hypoglycemia) and 192 females having Glucose Concentration > 140 (which is the case of Hyperglycemia)

1. Skin Thickness :-
    - It ranges from 7 - 99
    - It is a approximately positively skewed distribution having skewness = 0.6867941
    - The upper 25% of the data is more widely spread.
    - At skin thickness = 32mm maximum women are observed i.e., 31 women.
    - An outlier is present at 99mm.
    - 465 females have skin thickness > 30mm (condition for obesity)
    - Out of these 465 females 167 are the ones having number of pregnancies > 5
    - The result from skin thickness can be misleading as 227 observations are missing for this attribute.

1. Insulin Level :-
    - It ranges from 14 - 846
    - It is a highly positively skewed distribution having skewness = 2.149996
    - At insulin level of 105 mu U/ml maximum women are observed i.e., 11 females
    - The upper 25% of the data is more widely spread.
    - Outlier is present after 650 mu U/ml
    - 527 women have insulin level > 150 mu U/ml (case of insulin therapy)
    - The result from Serum insulin (mu U/ml) can be misleading as 527 observations are missing for this attribute.
    - A drastic fall in the frequency of women is observed with increasing values of Serum insulin (mu U/ml)

1. Body Mass Index :-
    - It ranges from 18.2 - 67.1
    - It is a moderately positively skewed distribution with skewness = 0.5916179

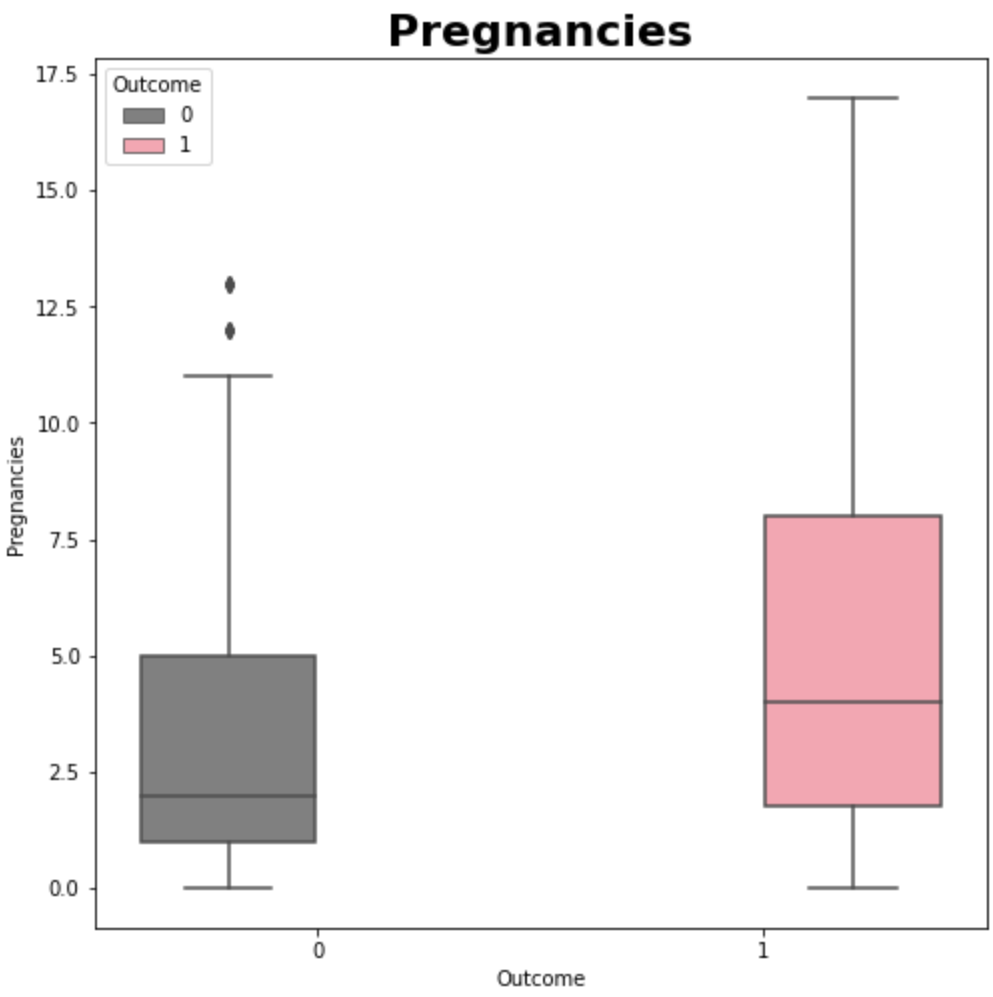
- At BMI = 32.0 maximum frequency of females is observed i.e., 13 women
- The upper 25% of the data is more widely spread.
- Outliers are present after BMI = 51 kg/m2
- 15 women are having BMI < 18.5 (condition of under weight)
- 662 women are having BMI > 24.9 (condition of over weight)

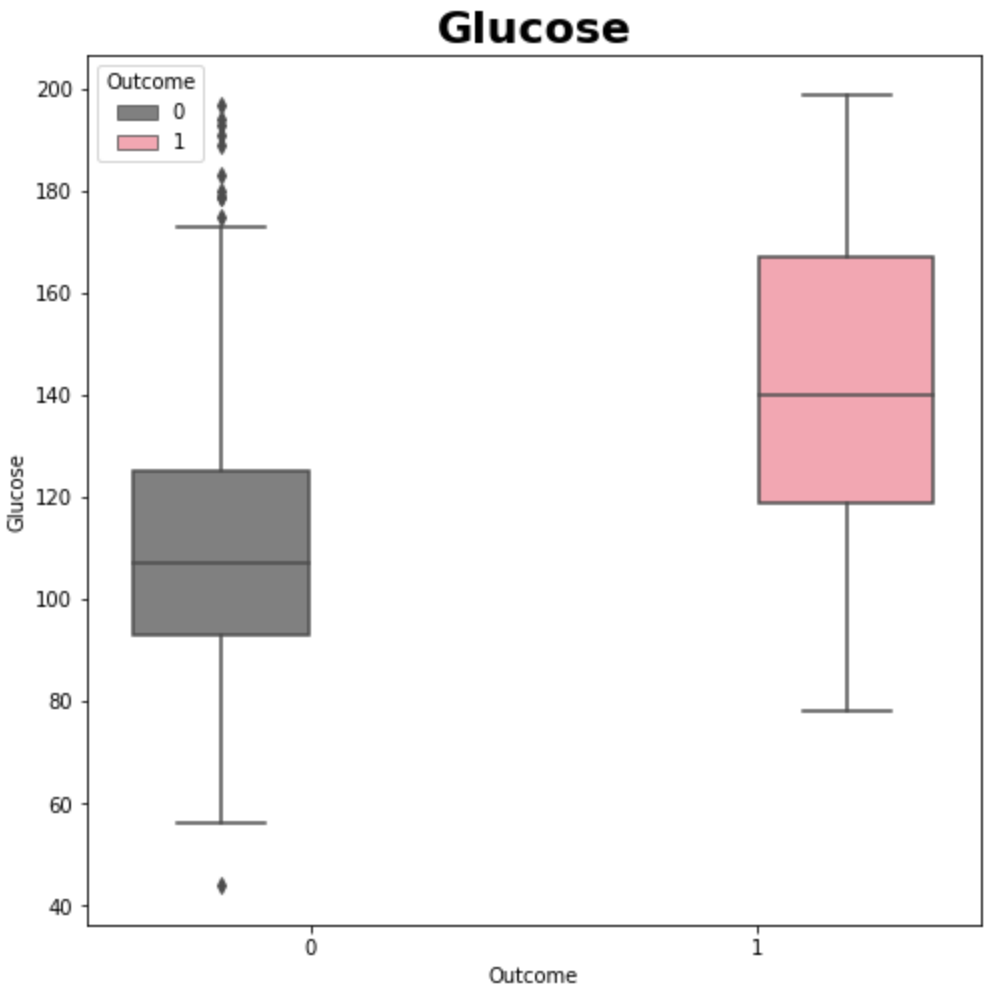
1. Diabetes Pedigree Function :-
    - It ranges from 0.78 - 2.42
    - It is a highly positively skewed distribution having skewness = 1.912418
    - As the value of Diabetes pedigree function increases the frequency of women decreases.
    - Maximum number of females are present at Diabetes pedigree function = 0.25 having 145 observations.
    - Only 10 women are observed at Diabetes pedigree function > 1.5
    - The upper 25% of the data is widely spread about the range of the Diabetes pedigree function.
    - Outliers are present after Diabetes pedigree function > 1.5

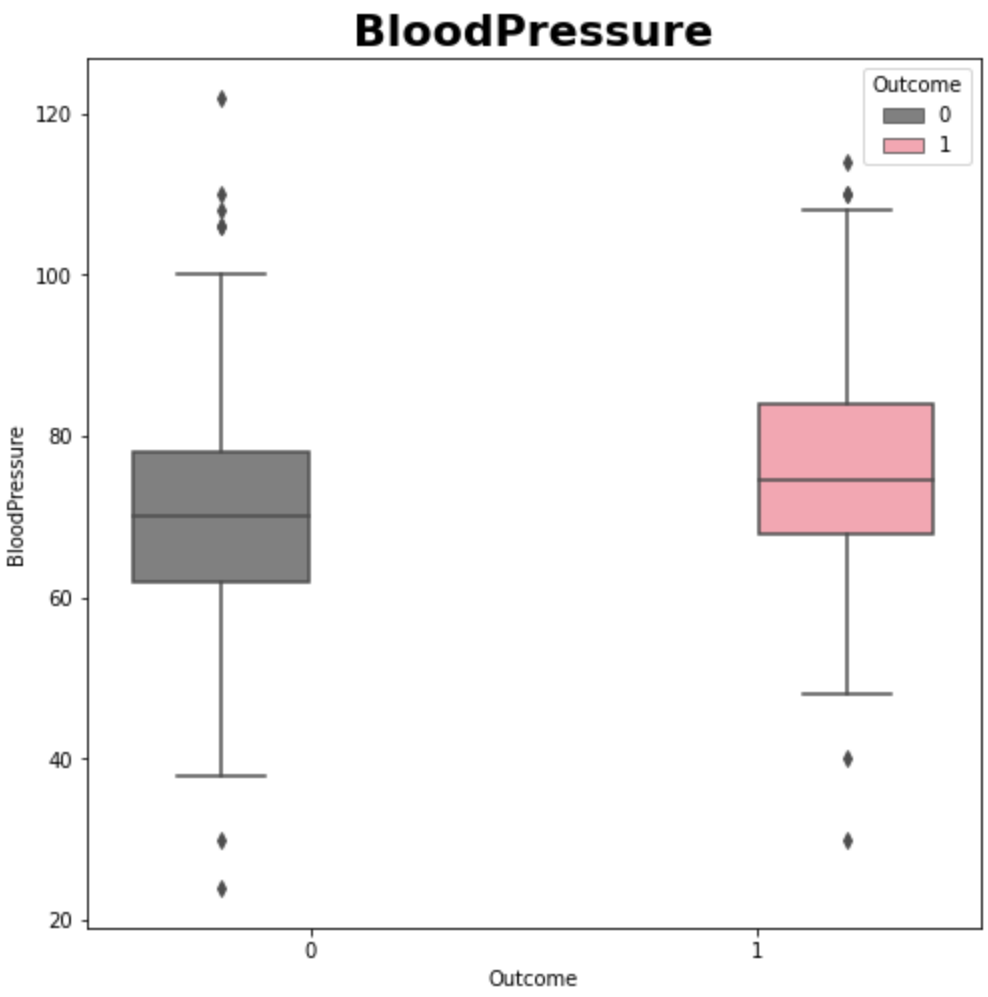
1. Age :-
    - It ranges from 21 - 81
    - It is a highly positively skewed distribution having skewness = 1.125188
    - The upper 25% of the data is widely spread about it's range.
    - As the age increases the frequency of women decreases.
    - The upper 25% of the data is widely spread about the range of the Diabetes pedigree function.
    - Outlier is present at age = 81
    - Maximum number of females are present at age = 22 having 72 observations.
    - 118 women are observed at age > 45

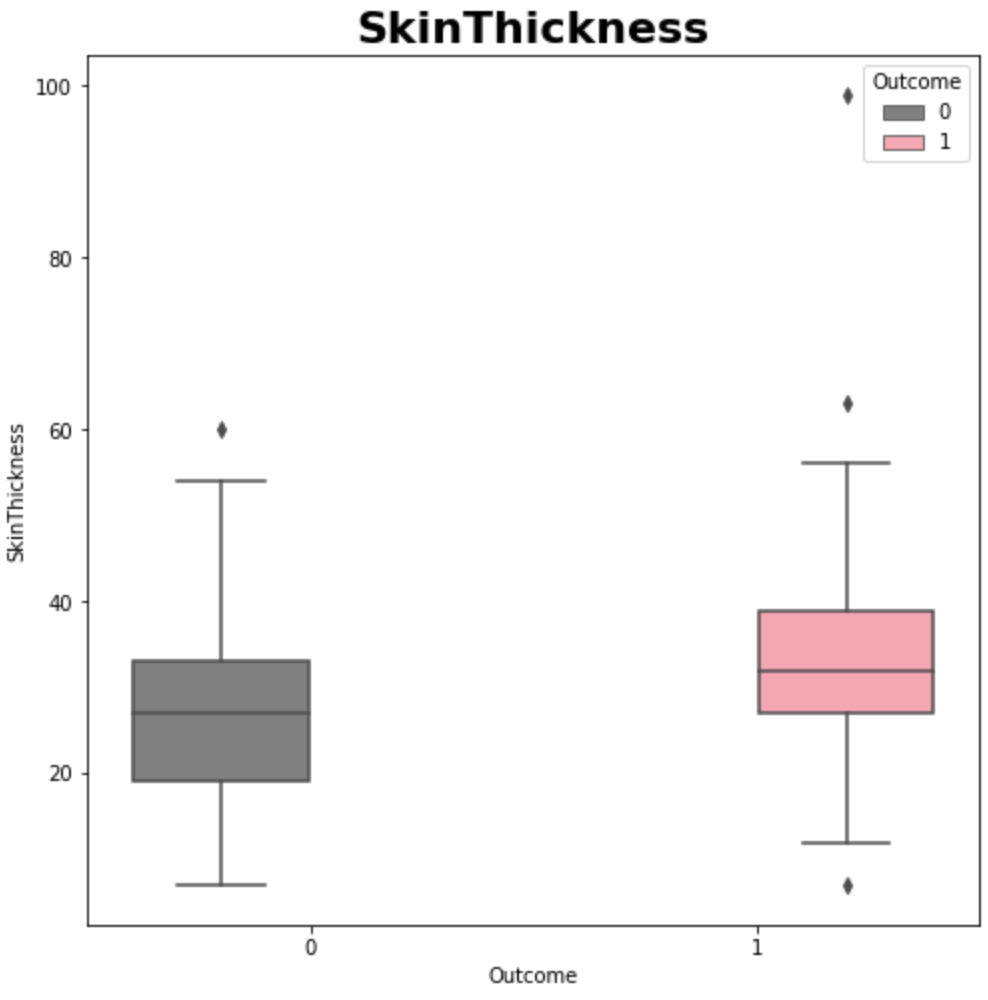
5. Bivariate Analysis

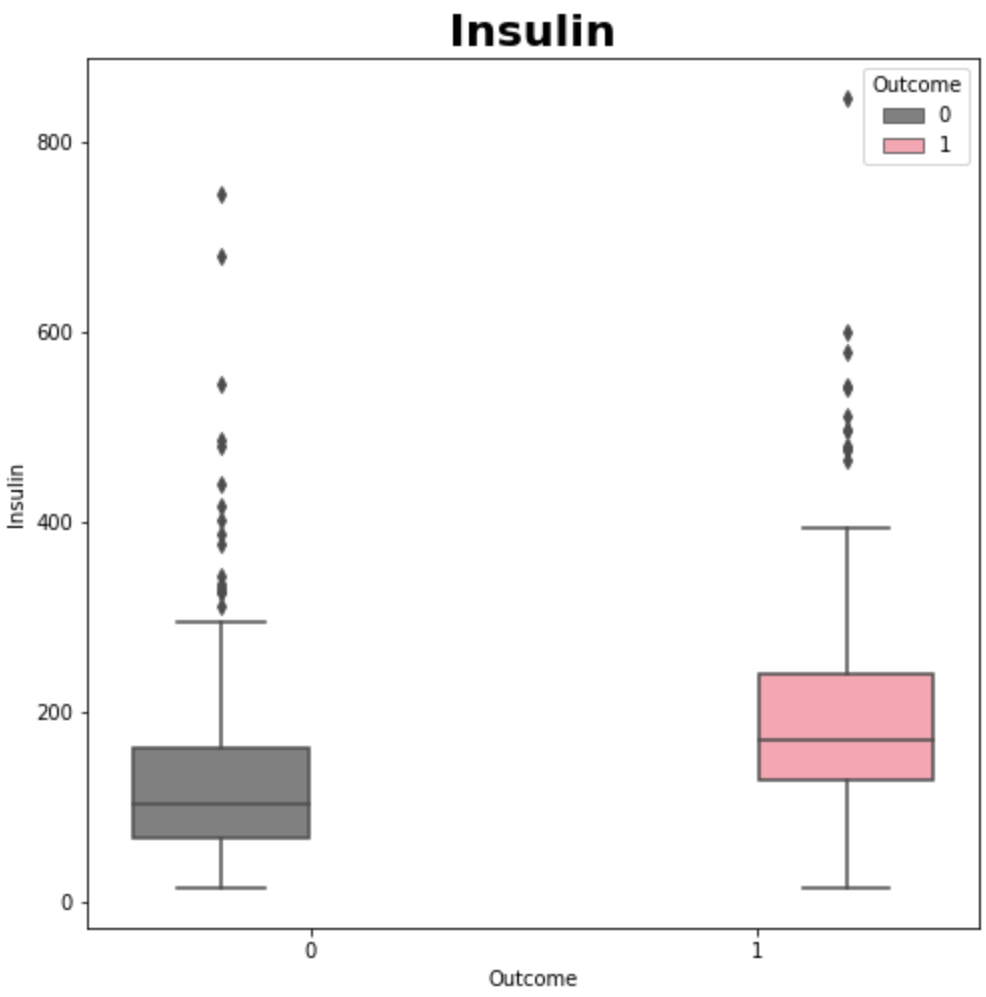
In [14]:
```
custom_colors = {0:'#808080', 1:'#ff9aa9'}
num_cols = len(df.columns)
for i in range(num_cols-1):
    x = df.iloc[:, i]
    name = x.name
    plt.figure(figsize=(8,8))
    sns.boxplot(y=x, x=df['Outcome'], hue=df['Outcome'], palette=custom_colors)
    plt.xlabel('Outcome')
    plt.title(name, fontweight='bold', fontsize=22)
    plt.show();
```

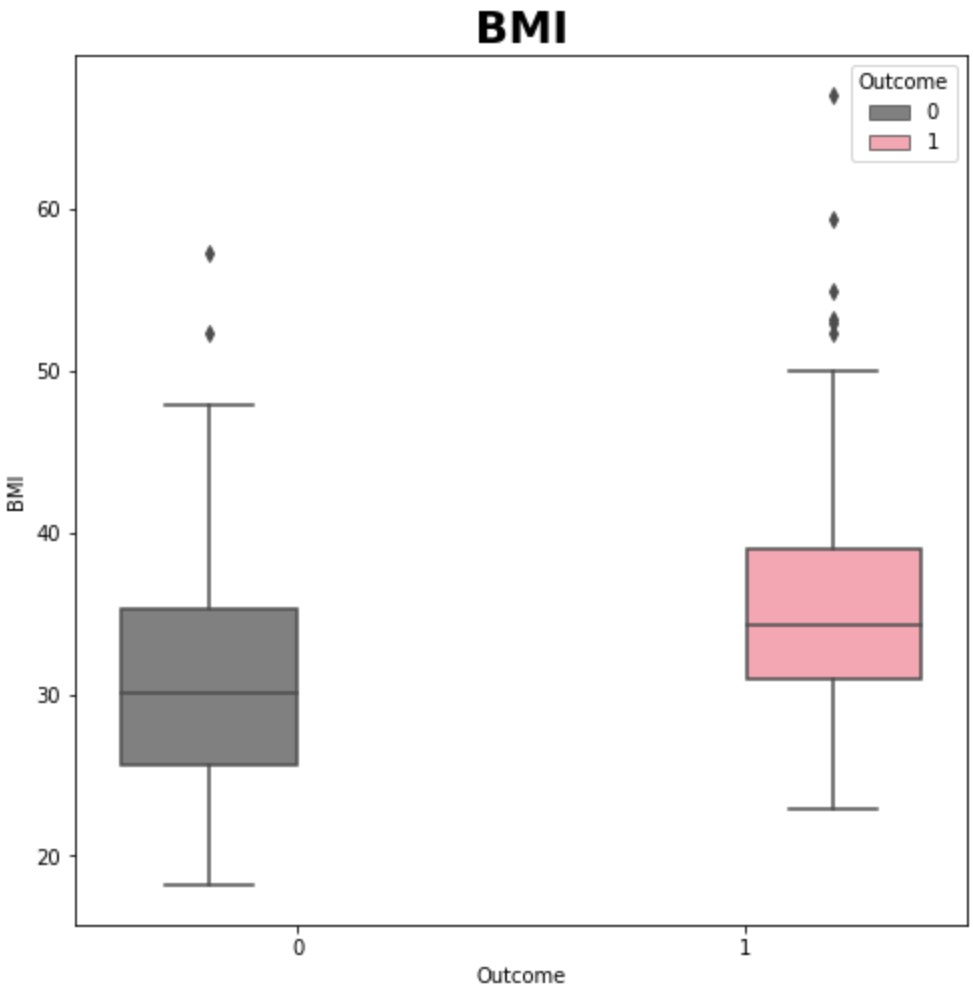

Pregnancies
Outcome
0
1
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Pregnancies
0
1
Outcome

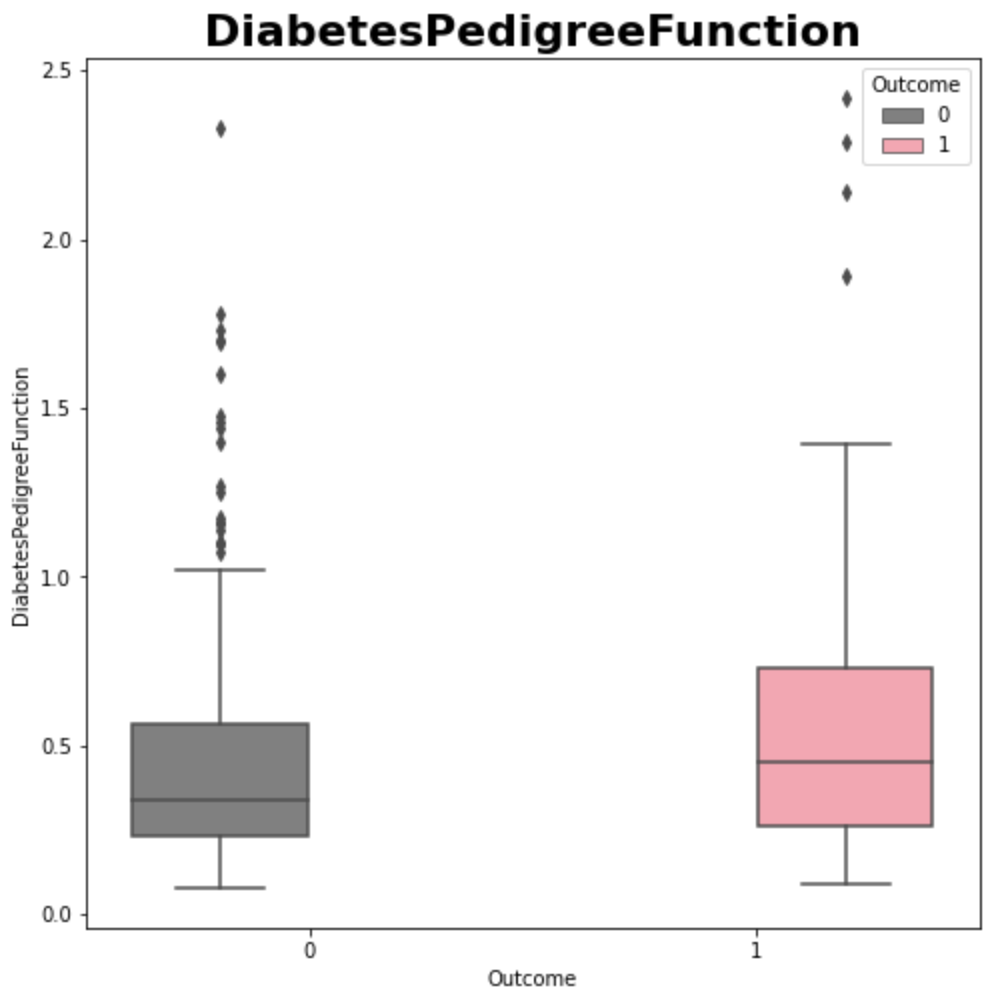

Glucose
Outcome
0
1
200
180
160
140
120
100
80
60
40
Glucose
0
1
Outcome

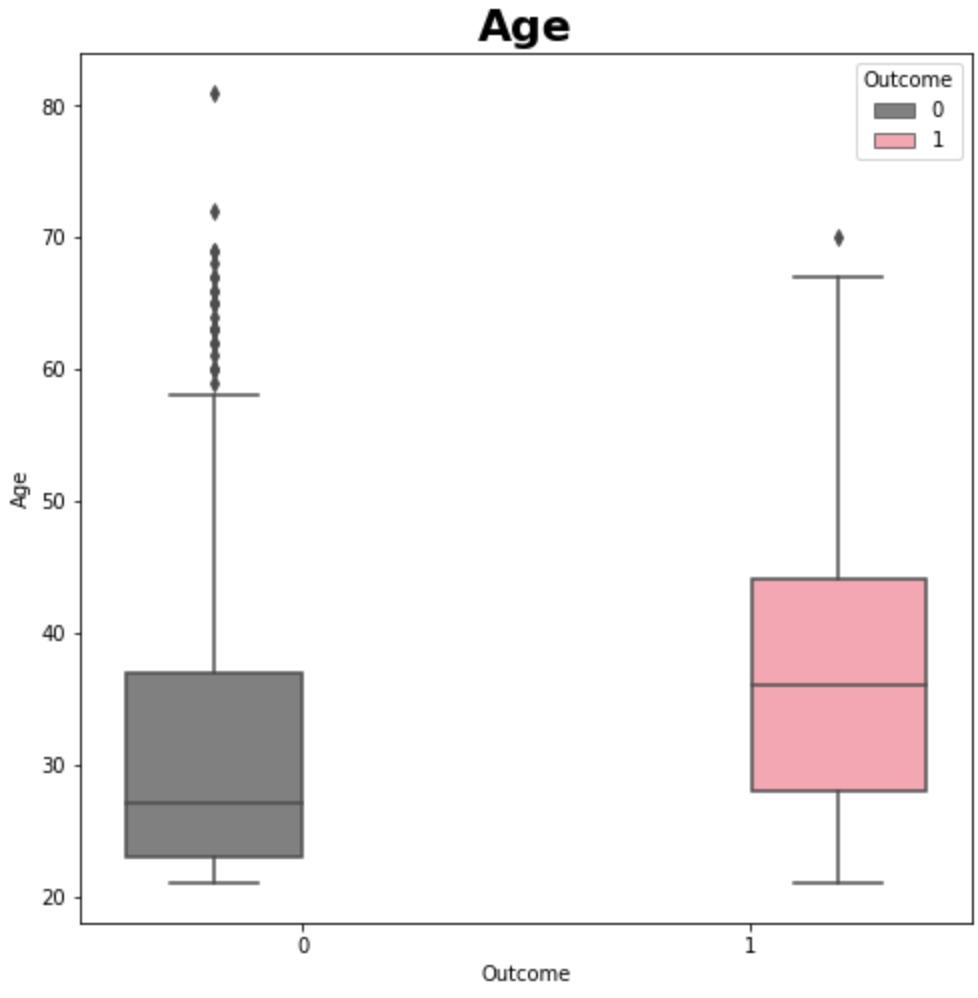

BloodPressure
Outcome
0
1
120
100
80
60
40
20
BloodPressure
0
1
Outcome


SkinThickness
Outcome
0
1
100
80
60
40
20
SkinThickness
0
1
Outcome


Insulin
Outcome
0
1
800
600
400
200
0
Insulin
0
1
Outcome


BMI
Outcome
0
1
60
50
40
30
20
BMI
0
1
Outcome

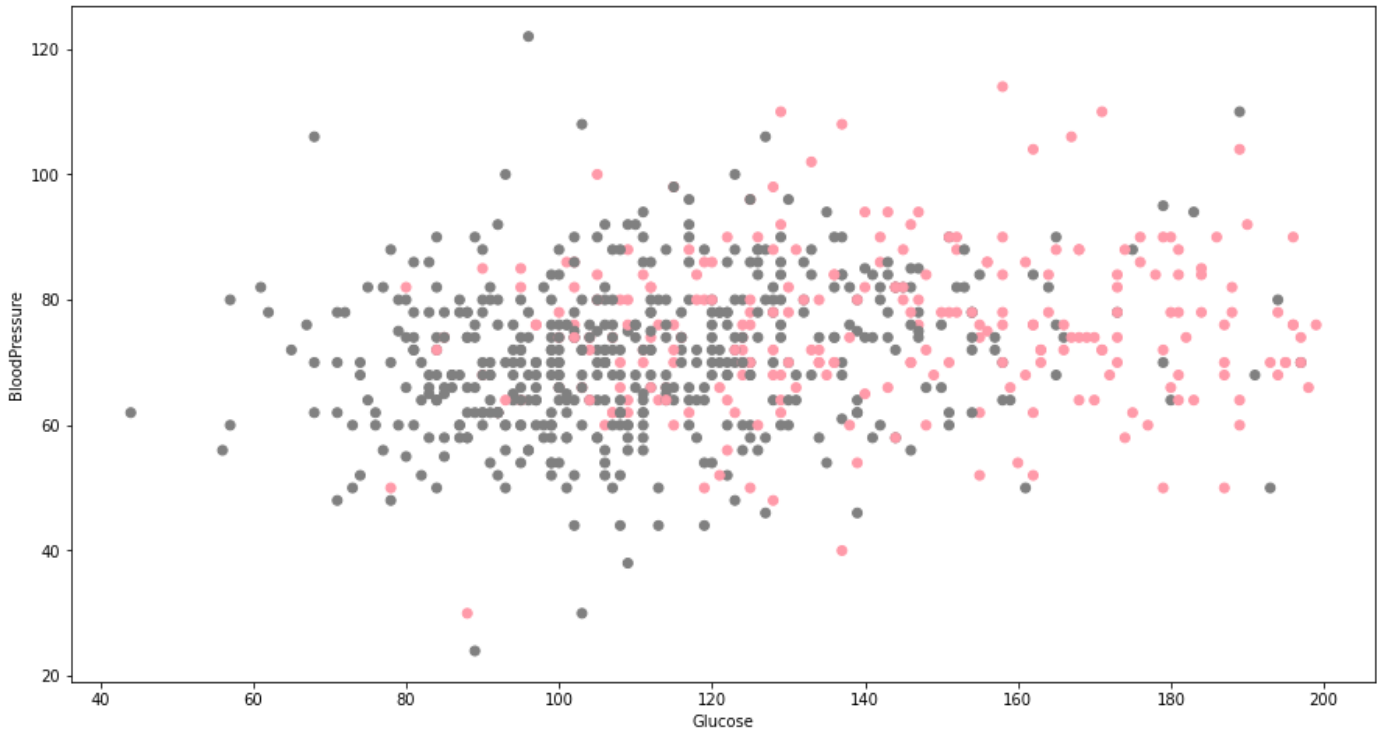

DiabetesPedigreeFunction
Outcome
0
1
DiabetesPedigreeFunction
2.5
2.0
1.5
1.0
0.5
0.0
0
1
Outcome


Age
Outcome
0
1
Age
80
70
60
50
40
30
20
0
1
Outcome

6. Multivariate Analysis

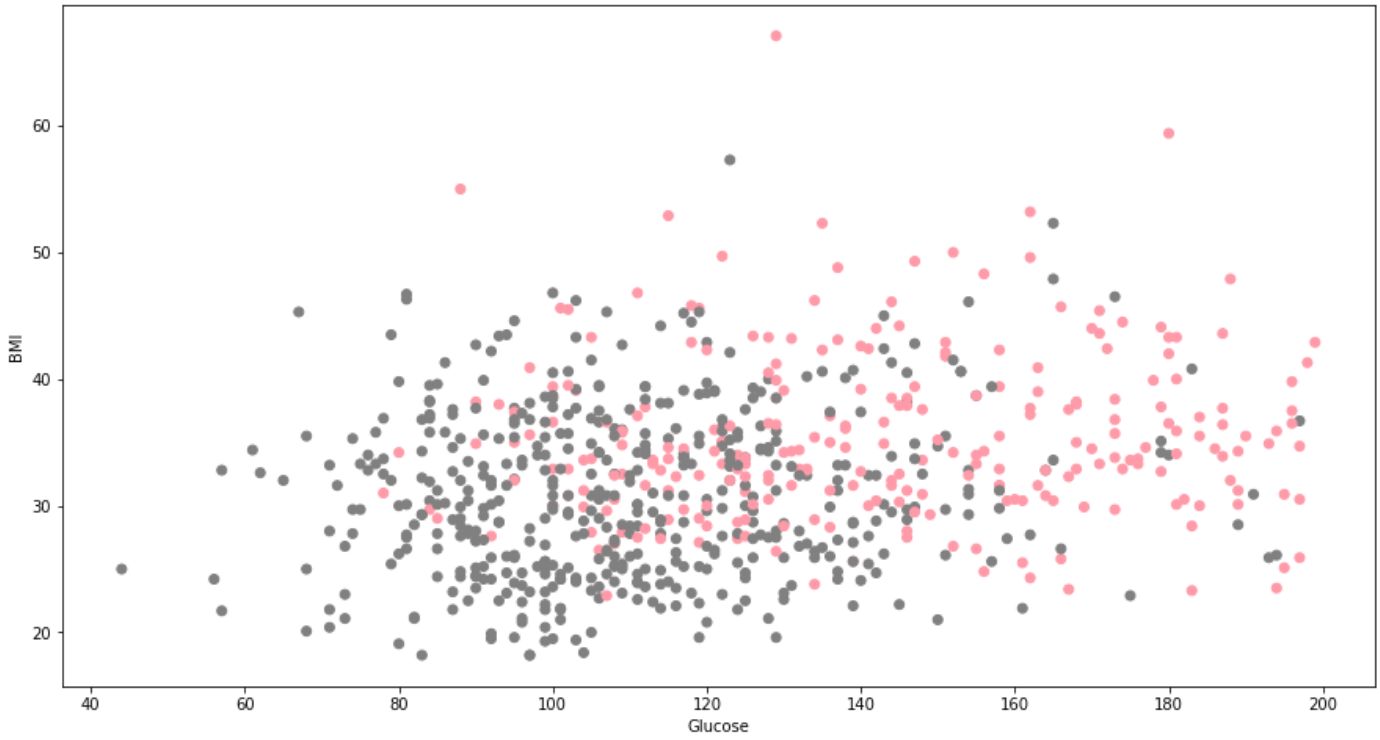
Considering only the most significant variables i.e., “Glucose Concentration”, “Body Mass Index” and “Blood Pressure”

In [15]:
```
plt.rcParams.update({'font.size': 10})
plt.rcParams["font.weight"] = "normal"
plt.figure(figsize=(15,8))
colors = {0:'#808080', 1:'#ff9aa9'}
plt.scatter(df['Glucose'], df['BloodPressure'], c=df['Outcome'].map(colors))
plt.xlabel("Glucose")
plt.ylabel("BloodPressure")
plt.show()
```



Women with Diabetes seems to have Higher Glucose concentration along with a Higher level of blood pressure.

In [16]:
```
plt.rcParams.update({'font.size': 10})
plt.rcParams["font.weight"] = "normal"
plt.figure(figsize=(15,8))
colors = {0:'#808080', 1:'#ff9aa9'}
plt.scatter(df['Glucose'], df['BMI'], c=df['Outcome'].map(colors))
plt.xlabel("Glucose")
plt.ylabel("BMI")
plt.show()
```



Women with Diabetes seems to have Higher Glucose concentration along with a higher body mass index (BMI).

7. Conclusion drawn from the EDA

- From the above graphs and hypothesis test, we can conclude that the top three most relevant features are “Glucose Concentration”, “Body Mass Index”, “Number of Pregnancies of females” and “Blood Pressure”.
- From the above graphs and hypothesis test, we can conclude that the variables skin thickness, Insulin level and Age are statiscally Non-significant.
- Non-Diabetic women seems to have lower levels of Insulin and Glucose as compared to women with Diabetes who recorded low to high levels of Insulin and high levels of Glucose.
- Non-Diabetic women seems to have lower/Normal levels of Blood Pressure as compared to women with Diabetes.

8. Modeling

Since the problem here is a binary classification problem therefore, following are the algorithms that will be considered for the further data analysis and prediction :

1. Logistic Regression
2. Decision Tree
3. Radom Forest

4. Principal Component Analysis (PCA)

8.1 Replacing 0's or NaN's with the median of the respective attribute :-

In [17]:
```
df = df.fillna(df.median())
df['Outcome'] = df['Outcome'].astype(int)
df.describe().T
```

Out[17]:

	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	121.656250	30.438286	44.000	99.75000	117.0000	140.25000	199.00
BloodPressure	768.0	72.386719	12.096642	24.000	64.00000	72.0000	80.00000	122.00
SkinThickness	768.0	29.108073	8.791221	7.000	25.00000	29.0000	32.00000	99.00
Insulin	768.0	140.671875	86.383060	14.000	121.50000	125.0000	127.25000	846.00
BMI	768.0	32.455208	6.875177	18.200	27.50000	32.3000	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

8.2 Scaling of the Data :-

- Min-max normalization is a normalization strategy which linearly transforms x to y = (x-min)/(max-min), where min and max are the minimum and maximum values in X, where X is the set of observed values of x.
- It can be easily seen that when x=min, then y=0, and When x=max, then y=1.

In [18]:
```
scaler = MinMaxScaler()
# fit and transform the data using the scaler
df = pd.DataFrame(scaler.fit_transform(df), columns=df.columns)
# print the normalized data
df.head()
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	C
0	0.352941	0.670968	0.489796	0.304348	0.133413	0.314928	0.234415	0.483333	
1	0.058824	0.264516	0.428571	0.239130	0.133413	0.171779	0.116567	0.166667	
2	0.470588	0.896774	0.408163	0.239130	0.133413	0.104294	0.253629	0.183333	
3	0.058824	0.290323	0.428571	0.173913	0.096154	0.202454	0.038002	0.000000	
4	0.000000	0.600000	0.163265	0.304348	0.185096	0.509202	0.943638	0.200000	

8.3 Splitting the dataset into 2 parts :-

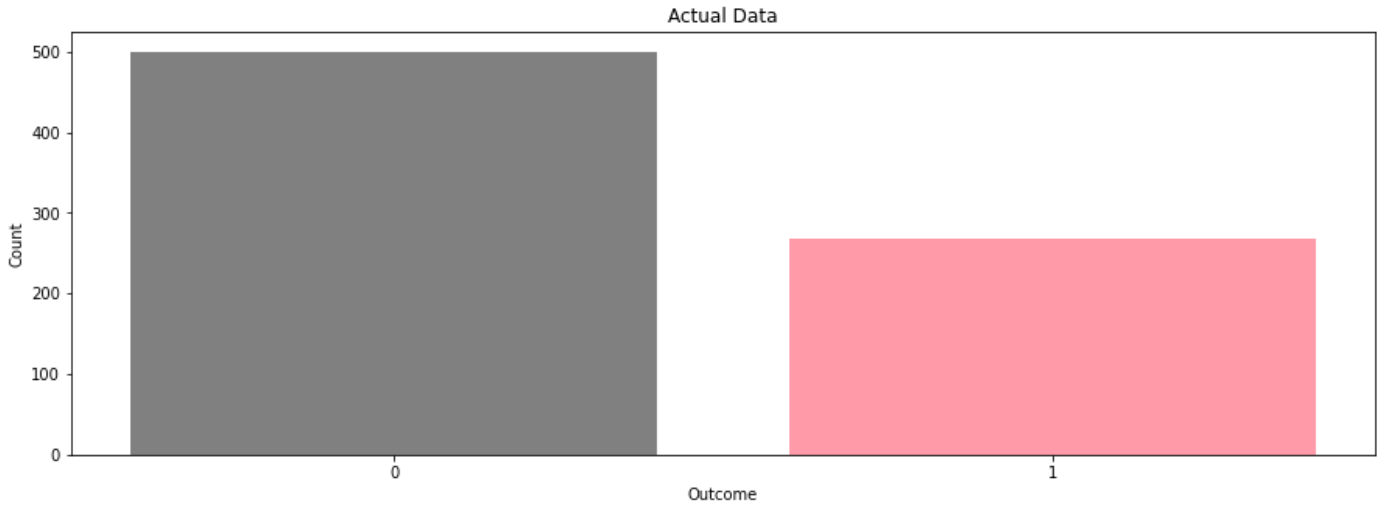
1. Training Dataset - The sample of data used to fit the model. The model sees and learns from this data.
2. Test Dataset - The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

In [19]:
```
column_sels = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'B
X = df.loc[:,column_sels]
y = df['Outcome'].astype(int)
```

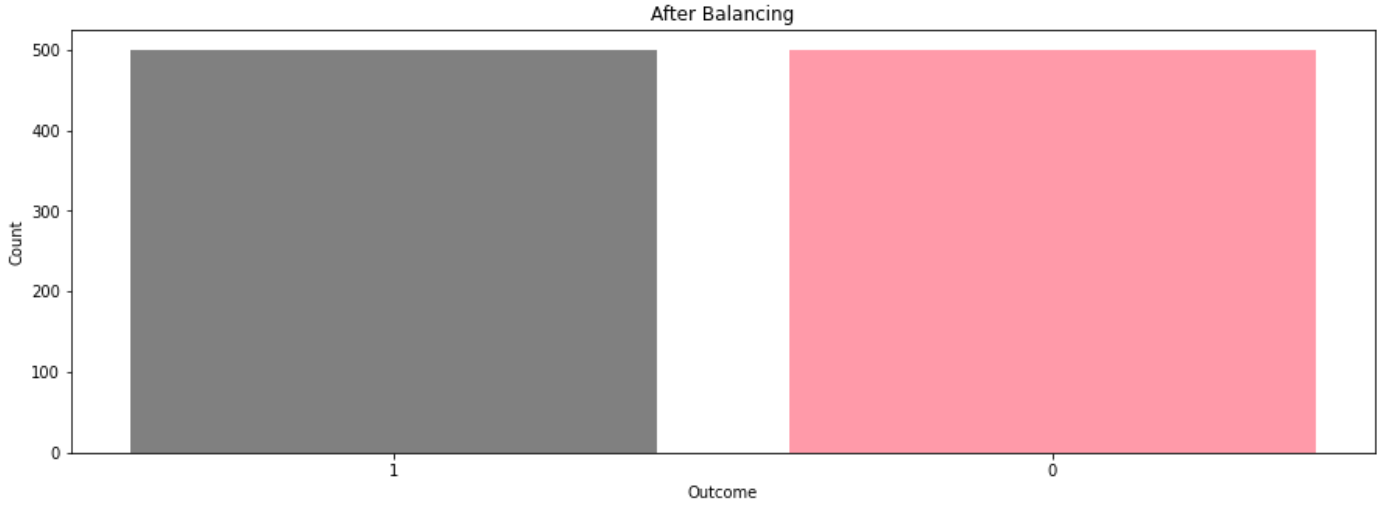
8.4 Class Balancing :-

- A balanced data set is a set that contains all elements observed in all time frame.
- Whereas unbalanced data is a set of data where certain years, the data category is not observed. Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally.
- Imbalanced data set will lead algorithms to get good results by returning the majority. That will be a problem if you are interested in the minority more. So, balancing is a way to force the algorithm to give more weight to the minority i.e., many classification learning algorithms have low predictive accuracy for the infrequent class.
- Oversampling and undersampling in data analysis are techniques used to adjust the class distribution of a data set

In [20]:
```
y2 = y.astype(str)
colors = {0:'#808080', 1:'#ff9aa9'}
plt.figure(figsize=(15,5))
plt.bar(list(y2.value_counts().index), y2.value_counts(),  color=['#808080','#ff9aa9'])
plt.title('Actual Data')
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.show();
```



In [21]:
```
oversample = SMOTE()
X, y = oversample.fit_resample(X, y)
y2 = y.astype(str)
colors = {0:'#808080', 1:'#ff9aa9'}
plt.figure(figsize=(15,5))
plt.bar(list(y2.value_counts().index), y2.value_counts(),  color=['#808080','#ff9aa9'])
plt.title('After Balancing')
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.show();
```



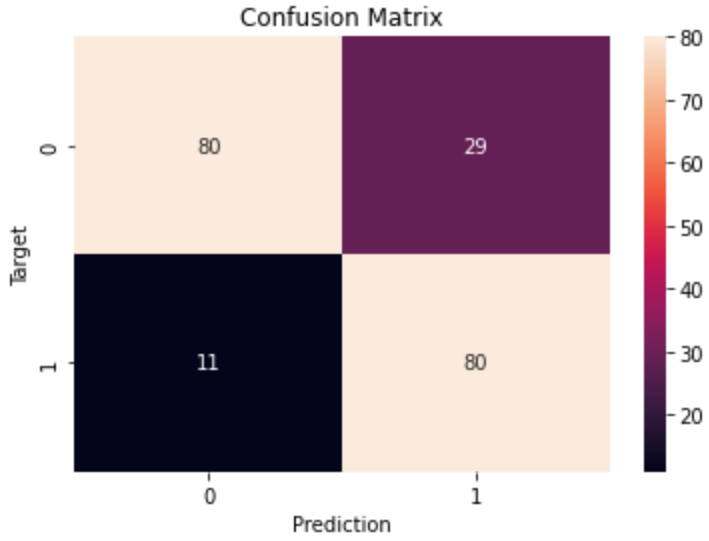
In [22]:
```
# using the train test split function
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=104,test_size=0.20,
```

9. Logistic Regression Model

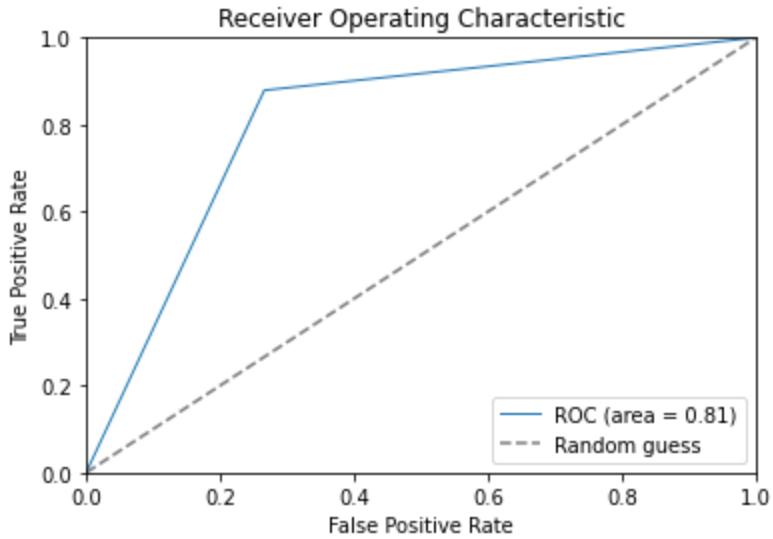
- Logistic Regression was used in the biological sciences in early twentieth century. It was then used in many social science applications. Logistic Regression is used when the dependent variable (target) is categorical.
- Here, I'm using Logistic Regression Model to predict whether a female is having diabetes (1) or not (0)
- It is also called Binary(dichotomous) Logistic Regression because the categorical response has only two 2 possible outcomes. Example: Diabetic (1) or Non-Diabetic (0)
- The outcome is measured by the following probabilistic link function called sigmoid due to its S-shaped.
- s(z) = 1/(1+e^{-z})

In [23]:
```
clf = LogisticRegression(random_state=None, solver='lbfgs')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
f1_score_val = f1_score(y_test,y_pred)
print("F1 Score: {:.2f}".format(f1_score_val))

cf = confusion_matrix(y_test, y_pred)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix');
```

```
Accuracy: 80.00%
F1 Score: 0.80
```



In [24]:
```
# compute ROC curve and area under the curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
# plot the ROC curve
plt.plot(fpr, tpr, lw=1, label='ROC (area = %0.2f)' % (roc_auc))
plt.plot([0, 1], [0, 1], '--', color='gray', label='Random guess')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



- ROC Curve :
  - ROC curves are commonly used to characterize the sensitivity/specificity tradeoffs for a binary classifier.
  - Here, true positive rate gives the sensitivity of the binary classifiers.
  - The false positive rate gives the specificity if the binary classifiers.
  - From the ROC Curve we choose 0.3 as the threshold value.

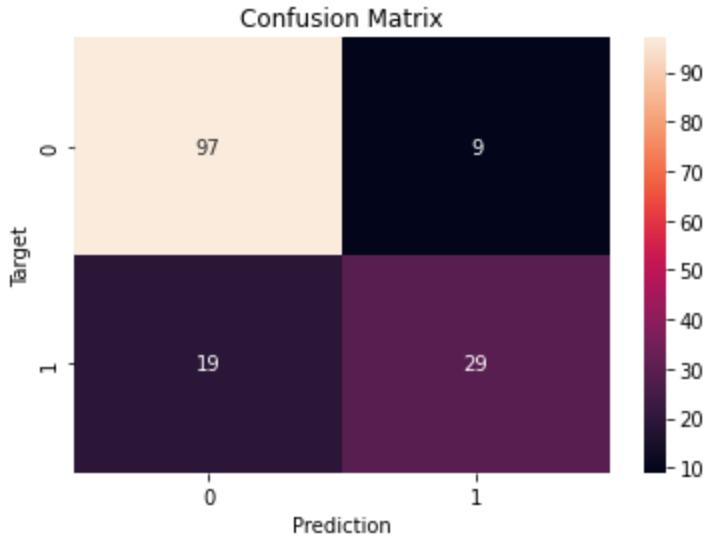
Removing Blood Pressure, Skin Thickness, Insulin and Age as they are least significant.

In [25]:
```
column_sels = ['Pregnancies', 'Glucose', 'BMI', 'DiabetesPedigreeFunction']
X = df.loc[:,column_sels]
```

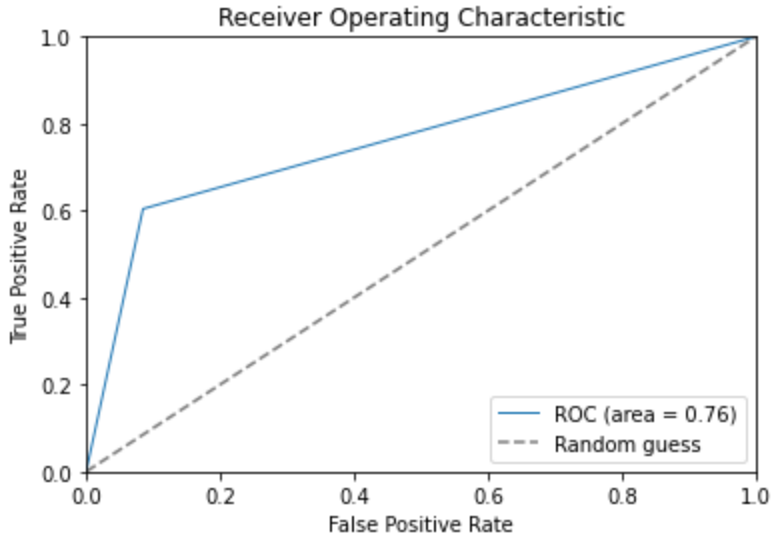
```
y = df['Outcome'].astype(int)
# using the train test split function
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=104,test_size=0.20,
```

In [26]:
```
clf = LogisticRegression(random_state=None, solver='lbfgs')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
f1_score_val = f1_score(y_test,y_pred)
print("F1 Score: {:.2f}".format(f1_score_val))

cf = confusion_matrix(y_test, y_pred)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix');
```

```
Accuracy: 81.82%
F1 Score: 0.67
```



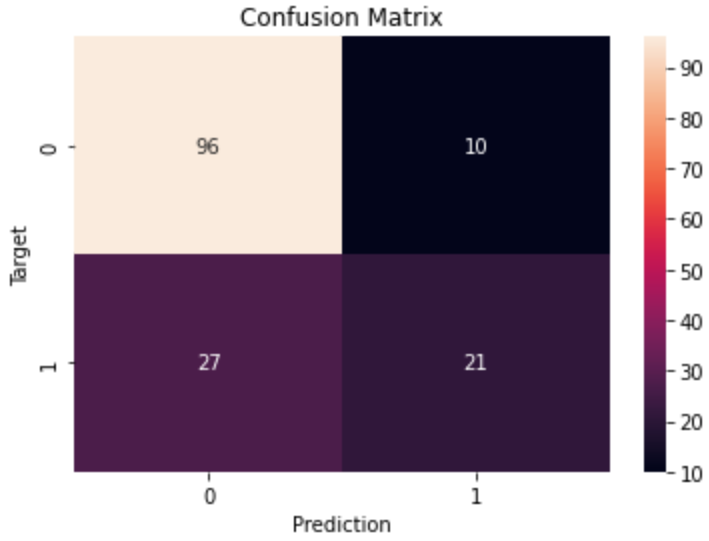
In [27]:
```
# compute ROC curve and area under the curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
# plot the ROC curve
plt.plot(fpr, tpr, lw=1, label='ROC (area = %0.2f)' % (roc_auc))
plt.plot([0, 1], [0, 1], '--', color='gray', label='Random guess')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



10. Decision Tree

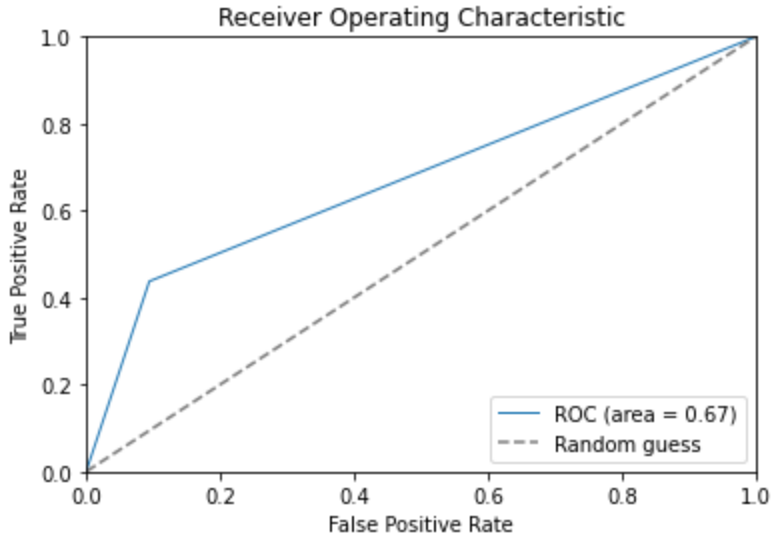
In [28]:
```
column_sels = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'B
X = df.loc[:,column_sels]
y = df['Outcome'].astype(int)
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=104,test_size=0.20,

clf = tree.DecisionTreeClassifier(max_depth=2, random_state=2, min_samples_leaf=10)
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
f1_score_val = f1_score(y_test,y_pred)
print("F1 Score: {:.2f}".format(f1_score_val))

cf = confusion_matrix(y_test, y_pred)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix');
```

```
Accuracy: 75.97%
F1 Score: 0.53
```



In [29]:
```
# compute ROC curve and area under the curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
```

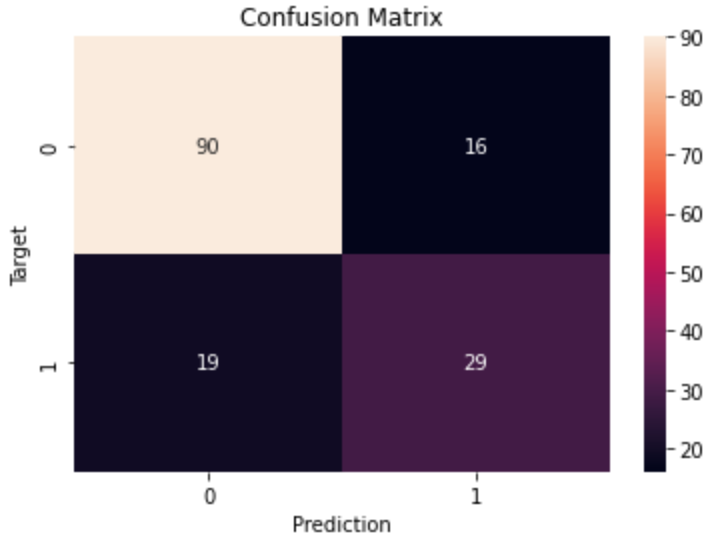
```
roc_auc = auc(fpr, tpr)
# plot the ROC curve
plt.plot(fpr, tpr, lw=1, label='ROC (area = %0.2f)' % (roc_auc))
plt.plot([0, 1], [0, 1], '--', color='gray', label='Random guess')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```



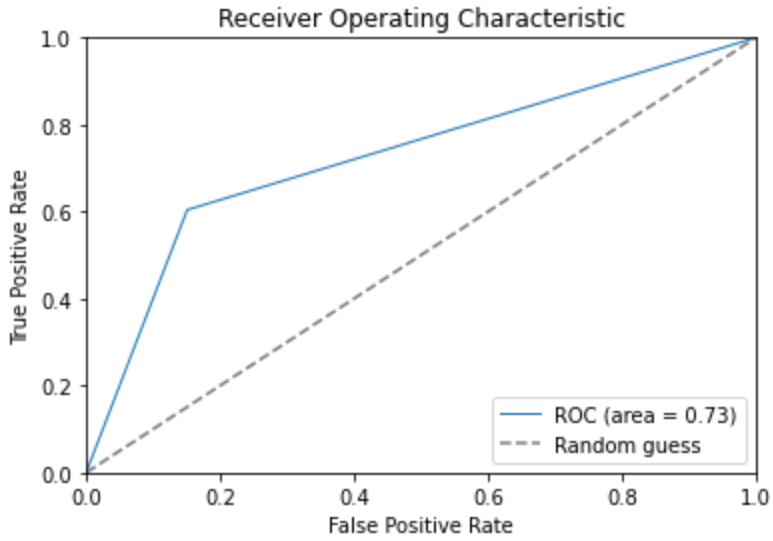
In [30]:

```
column_sels = ['Pregnancies', 'Glucose', 'BMI', 'DiabetesPedigreeFunction']
X = df.loc[:,column_sels]
y = df['Outcome'].astype(int)
# using the train test split function
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=104,test_size=0.20,

clf = tree.DecisionTreeClassifier(max_depth=6, random_state=10, min_samples_leaf=10, min
clf = clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
f1_score_val = f1_score(y_test,y_pred)
print("F1 Score: {:.2f}".format(f1_score_val))

cf = confusion_matrix(y_test, y_pred)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix');
```

```
Accuracy: 77.27%
F1 Score: 0.62
```



In [31]:
```
# compute ROC curve and area under the curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
# plot the ROC curve
plt.plot(fpr, tpr, lw=1, label='ROC (area = %0.2f)' % (roc_auc))
plt.plot([0, 1], [0, 1], '--', color='gray', label='Random guess')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

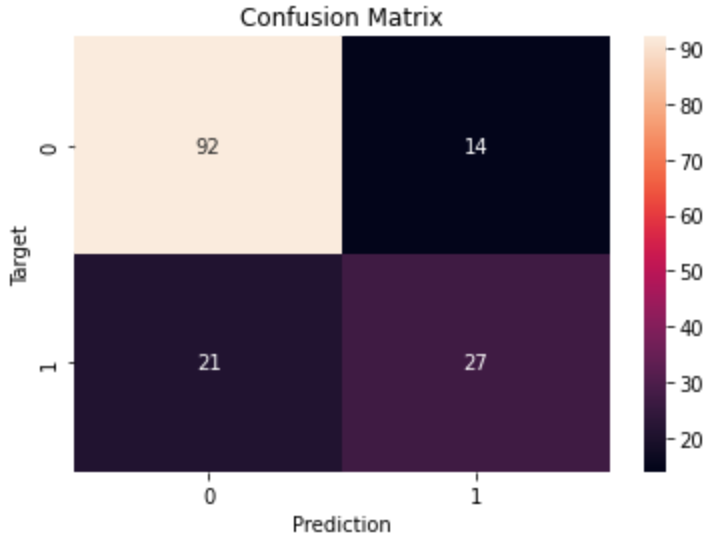


11. Random Forest

In [32]:
```
column_sels = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'B
X = df.loc[:,column_sels]
y = df['Outcome'].astype(int)
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=104,test_size=0.20,

rf = RandomForestClassifier(n_estimators = 200, max_depth = 4, max_features = 3, bootstr
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
```

```
f1_score_val = f1_score(y_test,y_pred)
print("F1 Score: {:.2f}".format(f1_score_val))

cf = confusion_matrix(y_test, y_pred)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix');
```

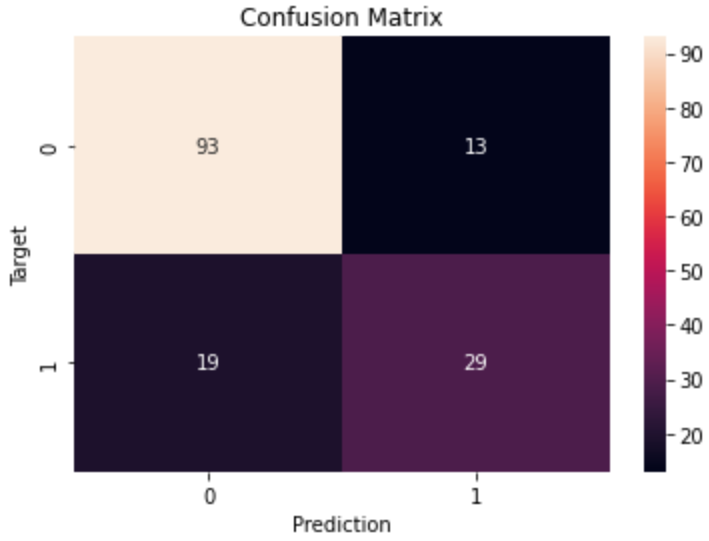
```
Accuracy: 77.27%
F1 Score: 0.61
```



In [33]:

```
column_sels = ['Pregnancies', 'Glucose', 'BMI', 'DiabetesPedigreeFunction']
X = df.loc[:,column_sels]
y = df['Outcome'].astype(int)
# using the train test split function
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=104,test_size=0.20,

rf = RandomForestClassifier(n_estimators = 200, max_depth = 4, max_features = 3, bootstr
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
f1_score_val = f1_score(y_test,y_pred)
print("F1 Score: {:.2f}".format(f1_score_val))

cf = confusion_matrix(y_test, y_pred)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix');
```

```
Accuracy: 79.22%
F1 Score: 0.64
```

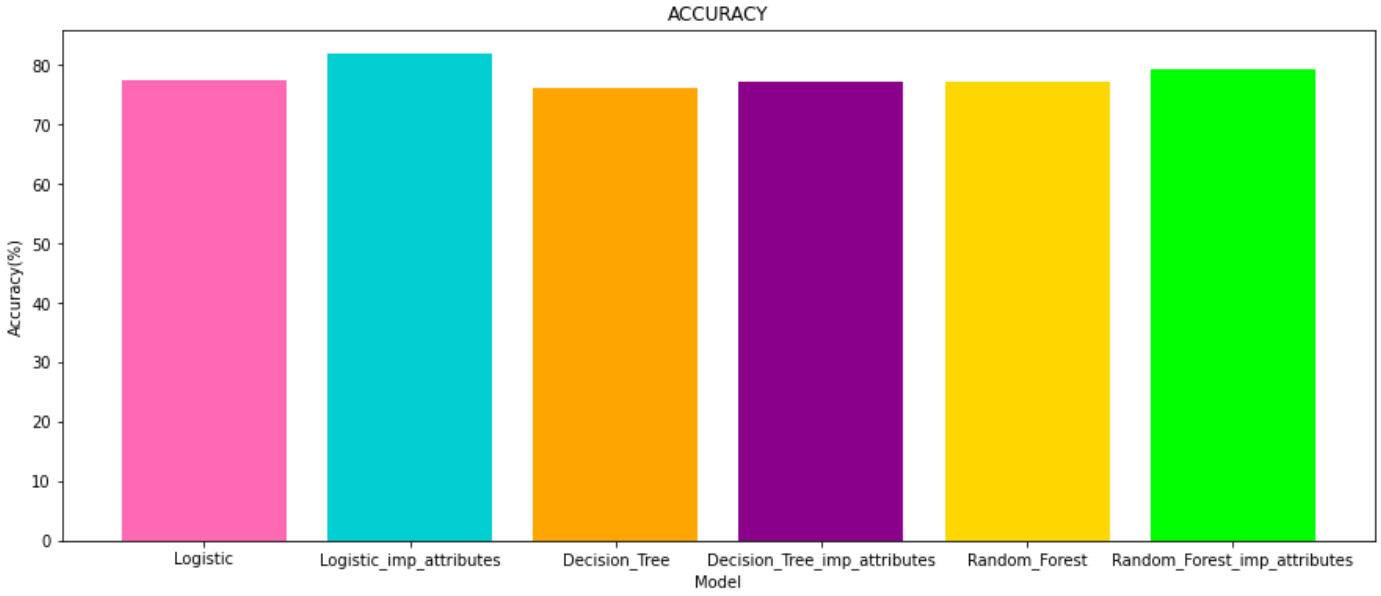


In [34]:
```
data_metrics = pd.DataFrame({'Model':['Logistic','Logistic_imp_attributes', 'Decision_Tr
             'Accuracy':[77.5, 81.82, 75.97, 77.27, 77.27, 79.22],
             'F1_Score':[0.76, 0.67, 0.53, 0.62, 0.61, 0.64],
             'False_Negative':[18, 19, 27, 19, 21, 19]}).set_index('Model')
data_metrics
```

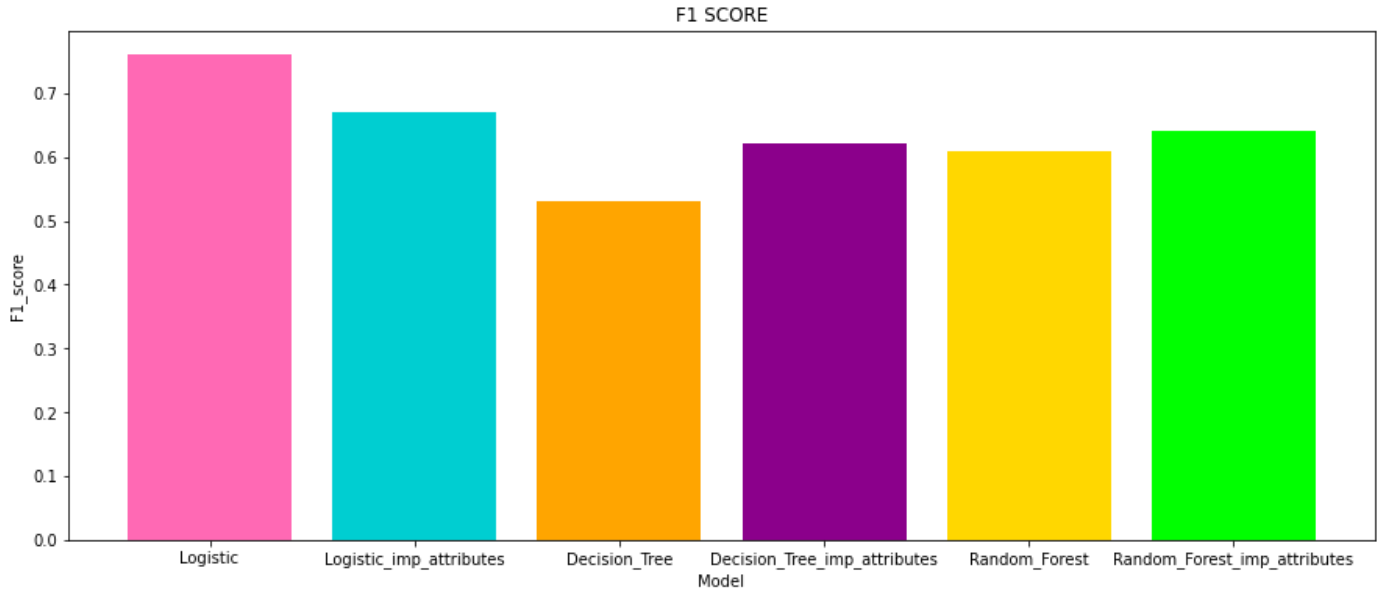
Out[34]:

Model	Accuracy	F1_Score	False_Negative
Logistic	77.50	0.76	18
Logistic_imp_attributes	81.82	0.67	19
Decision_Tree	75.97	0.53	27
Decision_Tree_imp_attributes	77.27	0.62	19
Random_Forest	77.27	0.61	21
Random_Forest_imp_attributes	79.22	0.64	19

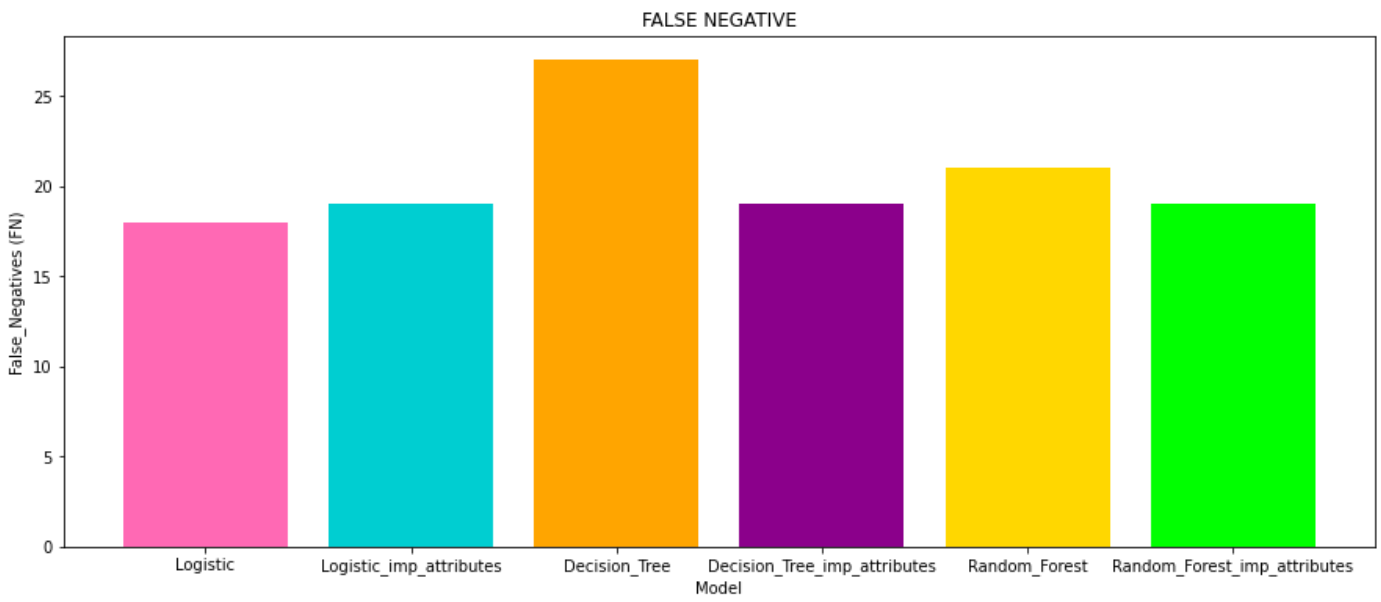
In [35]:
```
plt.figure(figsize=(15,6))
c = ['#FF69B4', '#00CED1', '#FFA500', '#8B008B', '#FFD700', '#00FF00']
plt.bar(list(data_metrics.index), data_metrics.Accuracy, color=c)
plt.title('ACCURACY')
plt.xlabel('Model')
plt.ylabel('Accuracy(%)')
plt.show();
```



In [36]:
```
plt.figure(figsize=(15,6))
c = ['#FF69B4', '#00CED1', '#FFA500', '#8B008B', '#FFD700', '#00FF00']
plt.bar(list(data_metrics.index), data_metrics.F1_Score, color=c)
plt.title('F1 SCORE')
plt.xlabel('Model')
plt.ylabel('F1_score')
plt.show();
```



In [37]:
```
plt.figure(figsize=(15,6))
c = ['#FF69B4', '#00CED1', '#FFA500', '#8B008B', '#FFD700', '#00FF00']
plt.bar(list(data_metrics.index), data_metrics.False_Negative, color=c)
plt.title('FALSE NEGATIVE')
plt.xlabel('Model')
plt.ylabel('False_Negatives (FN)')
plt.show();
```



12. PCA

- In simple words, principal component analysis is a method of extracting important variables (in form of components) from a large set of variables available in a data set. It extracts low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible. With fewer variables, visualization also becomes much more meaningful.
- PCA is more useful when dealing with 3 or higher dimensional data.
- Principal Component Analysis (PCA) is a useful technique for exploratory data analysis, allowing you to better visualize the variation present in a dataset with many variables.
- It is particularly helpful in the case of “wide” datasets, where you have many variables for each sample.
- PCA allows you to see the overall “shape” of the data, identifying which samples are similar to one another and which are very different.
- Basics of PCA are as follows: you take a dataset with many variables, and you simplify that dataset by turning your original variables into a smaller number of “Principal Components”.

12.1 Performing PCA :-

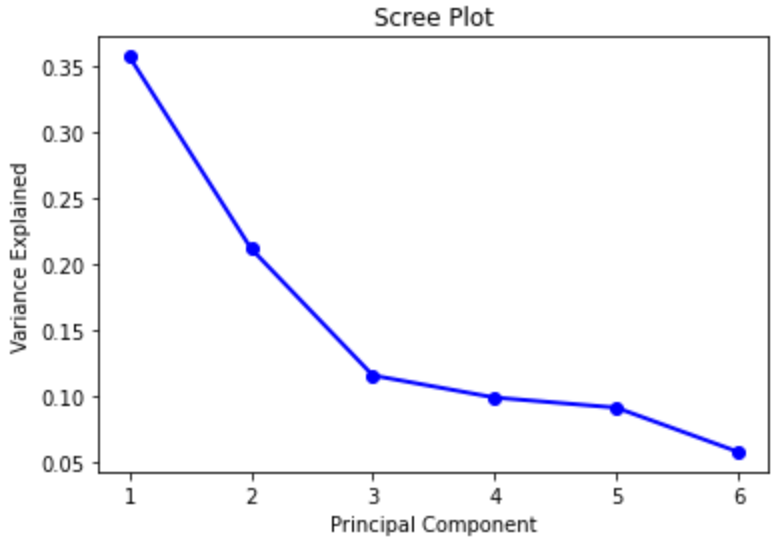
In [38]:
```
column_sels = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'B
X = df.loc[:,column_sels]
y = df['Outcome'].astype(int)
pca = PCA(n_components=6)
X = pca.fit_transform(X)
explained_variance = pca.explained_variance_ratio_
print(explained_variance)
print(explained_variance.sum()*100)
```

```
[0.356716   0.21134494 0.11539223 0.09864849 0.09098717 0.05755619]
93.06450182305748
```

12.2 Scree Plot And Cumulative Variance Curve :-

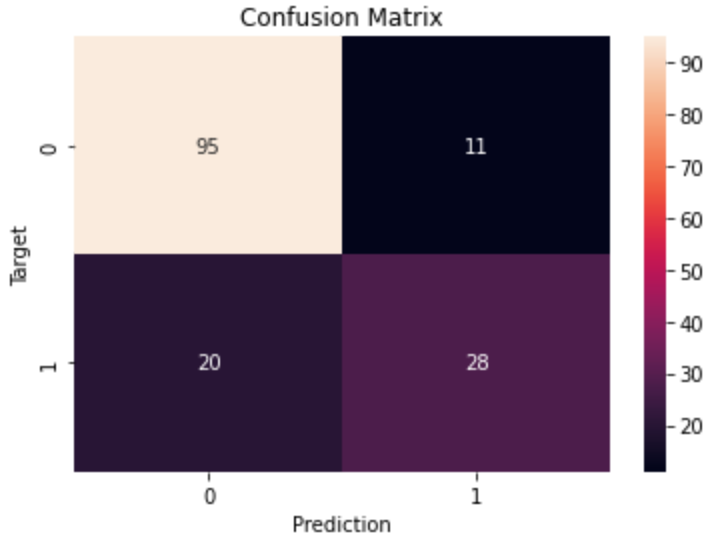
- To determine what should be an 'ideal' set of features we should take after using PCA, we use a screeplot diagram.

In [39]:
```
#fit PCA model to data
PC_values = np.arange(pca.n_components_) + 1
plt.plot(PC_values, pca.explained_variance_ratio_, 'o-', linewidth=2, color='blue')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```



In [40]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=104,test_size=0.20,
clf = LogisticRegression(random_state=2, solver='lbfgs')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)


accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: {:.2f}%".format(accuracy * 100))
f1_score = f1_score(y_test,y_pred)
print("F1 Score: {:.2f}".format(f1_score))

cf = confusion_matrix(y_test, y_pred)
plt.figure()
sns.heatmap(cf, annot=True)
plt.xlabel('Prediction')
plt.ylabel('Target')
plt.title('Confusion Matrix');
```

```
Accuracy: 79.87%
F1 Score: 0.64
```



CONCLUSION

1. It can be concluded that Logistic model is giving the best performance among all the three models with or without using PCA.
2. Logistic model without using PCA is choosen as the best model for this dataset as it gives higher accuracy and least no. of type-2 errors.

- Accuracy :
  - Logistic model = 81.82%
  - Logistic model (with PCA) = 79.87%
- Type-2 Errors :
  - Logistic model = 19
  - Logistic model (with PCA) = 20