In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [2]:
```
data = pd.read_csv(r"C:\Users\varun\OneDrive\Desktop\Project 2 - Diabetes Data\P
```

Exploratory data analysis

In [3]:
```
data.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFun
0	6	148	72	35	0	33.6	0.627
1	1	85	66	29	0	26.6	0.351
2	8	183	64	0	0	23.3	0.672
3	1	89	66	23	94	28.1	0.167
4	0	137	40	35	168	43.1	2.288

In [4]:
```
data.describe()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	Diabete
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000(
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 [5]:
```
data.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 [6]:
```
data.shape
```

```
(768, 9)
```

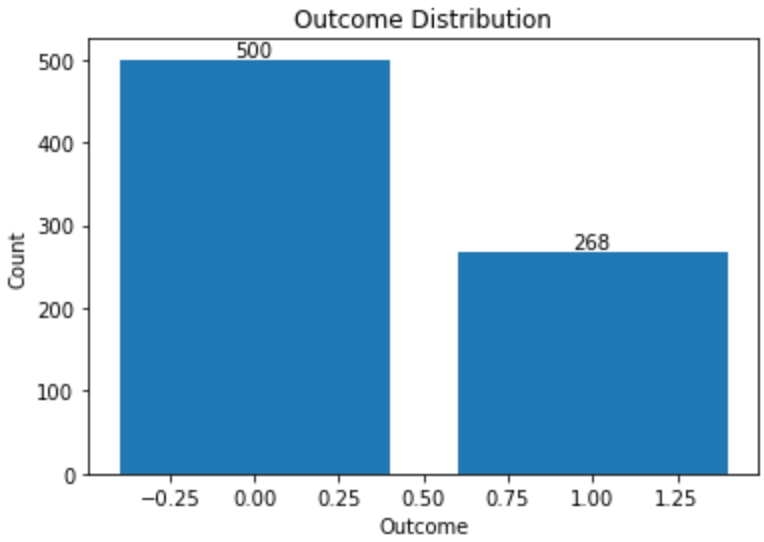
Checking for missing values

In [7]:
```
data.isnull().sum()
```

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

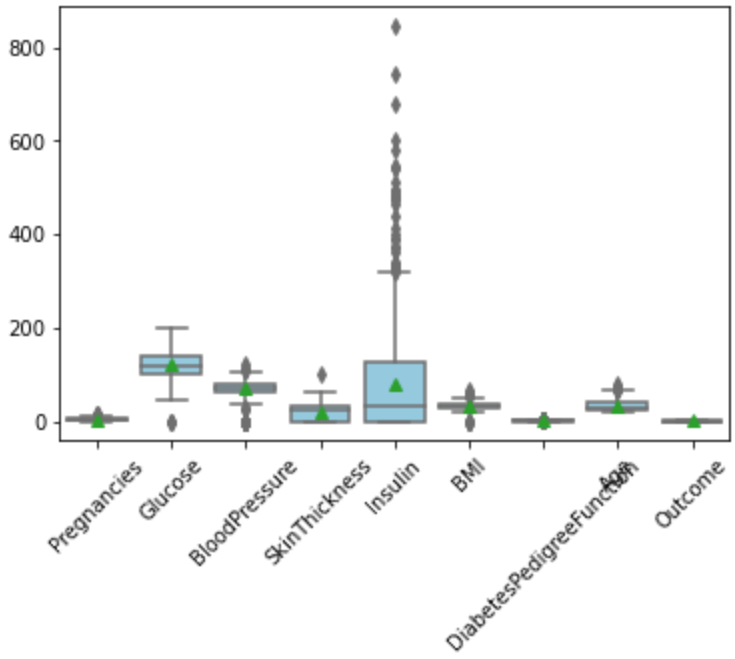
Our data has no missing values

In [8]:
```
value_counts = data['Outcome'].value_counts()

# Create a bar plot
plt.bar(value_counts.index, value_counts.values)

# Set labels and title
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.title('Outcome Distribution')

for x, y in zip(value_counts.index, value_counts.values):
    plt.text(x, y, str(y), ha='center', va='bottom')

# Show the plot
plt.show()
```



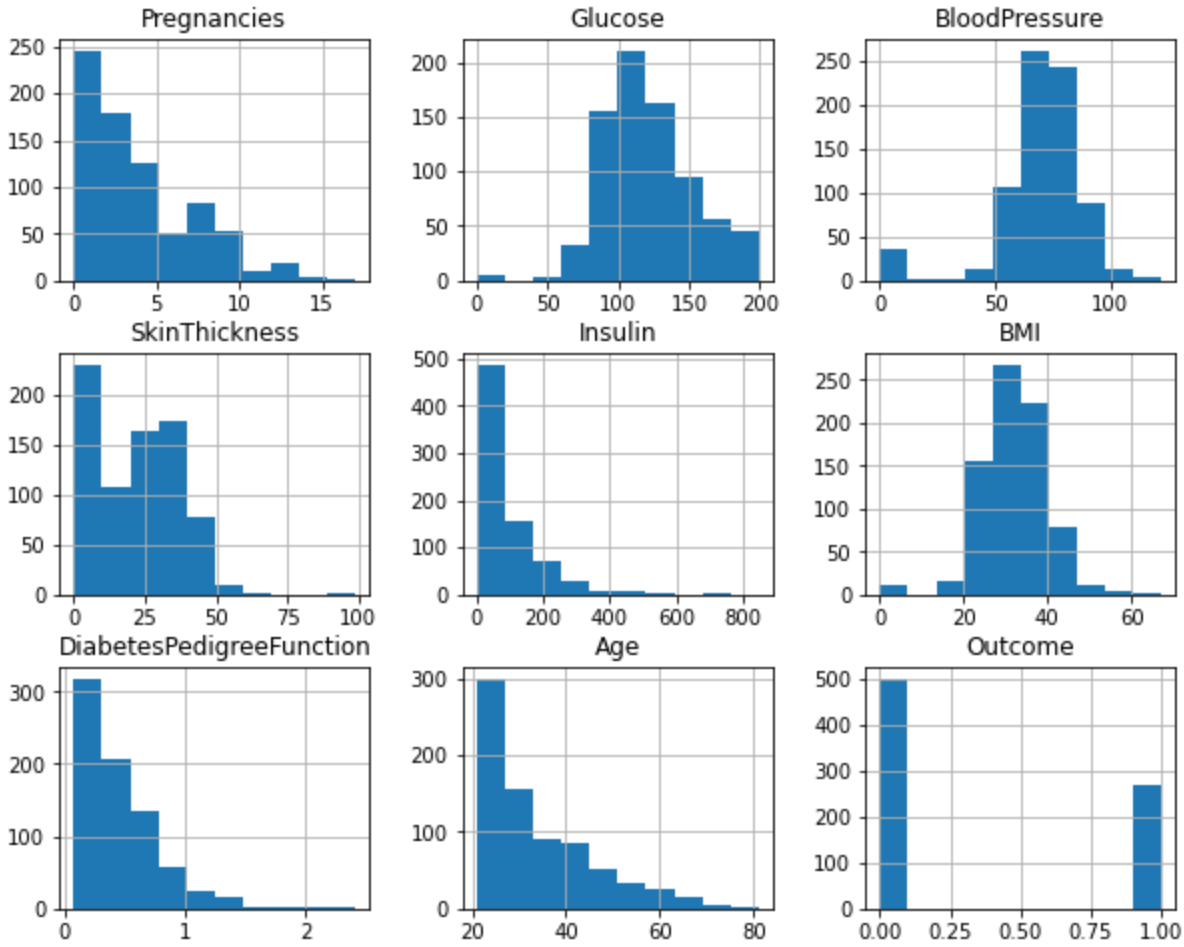
Checking the Outliers

In [9]:
```
sns.boxplot(
    data=data,
    orient="v",
    showmeans=True,
    color='skyblue',
)
plt.xticks(rotation=45)
plt.show()
```



In [11]:
```
data.hist(figsize = (10,8))
plt.show
```

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

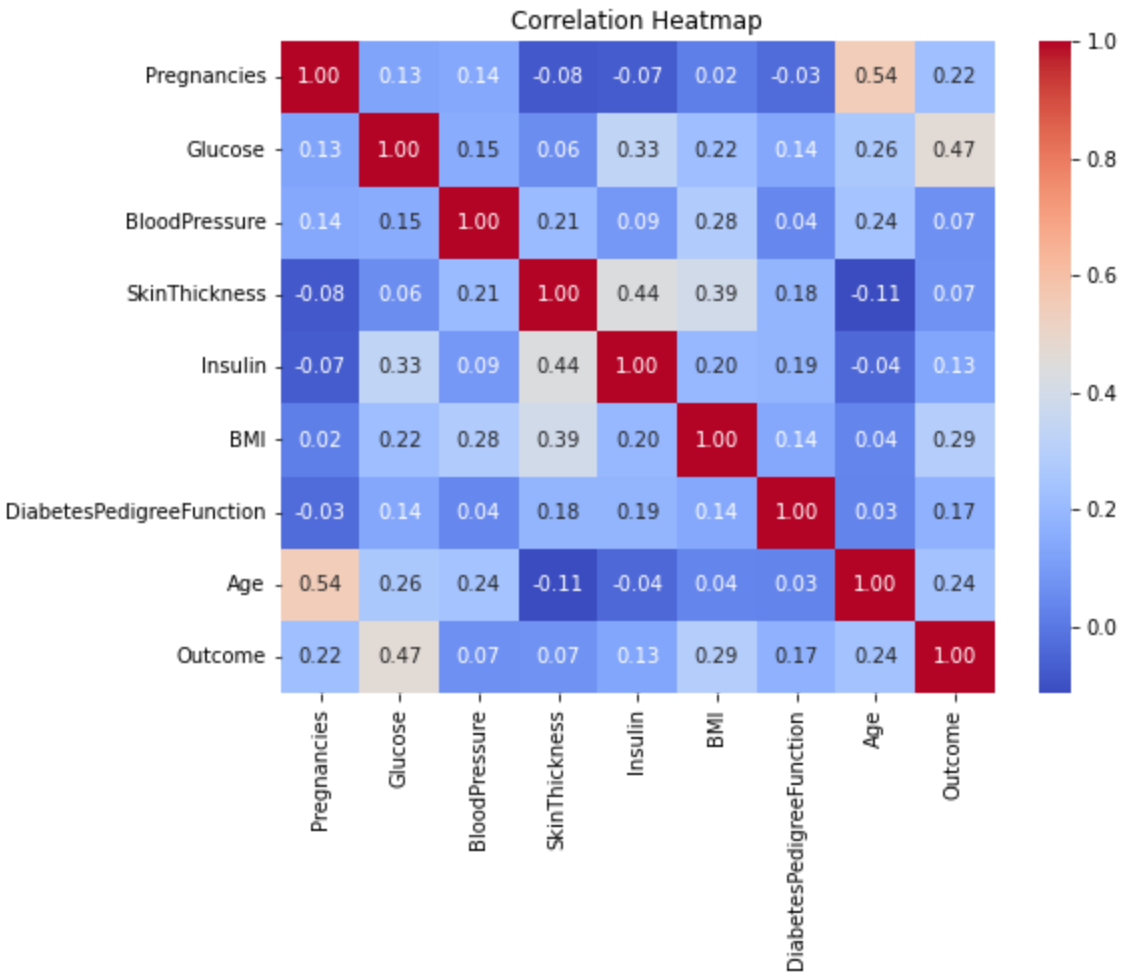


Checking Correlations

In [12]:
```
correlation = data.corr()
correlation
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.01
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.22
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.28
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.39
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.19
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.00
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.14
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.03
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.29

In [13]:
```
plt.figure(figsize=(8, 6))  # Set the figure size (optional)
sns.heatmap(correlation, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()
```



- Pregnancies is positively correlated with glucose and insulin, meaning that wo pregnancies tend to have higher blood glucose levels and insulin levels.
- Glucose is positively correlated with blood pressure, BMI, and diabetes pedigr people with higher blood glucose levels tend to have higher blood pressure, h genetic risk of diabetes.
- Blood pressure is positively correlated with BMI and diabetes pedigree functio high blood pressure tend to have higher BMI and a higher genetic risk of diab
- Skin thickness is positively correlated with BMI and diabetes pedigree functior thicker skin tend to have higher BMI and a higher genetic risk of diabetes.
- Insulin is positively correlated with BMI and diabetes pedigree function, meani insulin levels tend to have higher BMI and a higher genetic risk of diabetes.
- BMI is positively correlated with diabetes pedigree function, meaning that peo have a higher genetic risk of diabetes.

- Age is positively correlated with glucose and blood pressure, meaning that olc blood glucose levels and blood pressure.
- Outcome (diabetes) is positively correlated with glucose, blood pressure, BMI and age, meaning that people with higher blood glucose levels, blood pressur to have diabetes.

Removing the Outliers

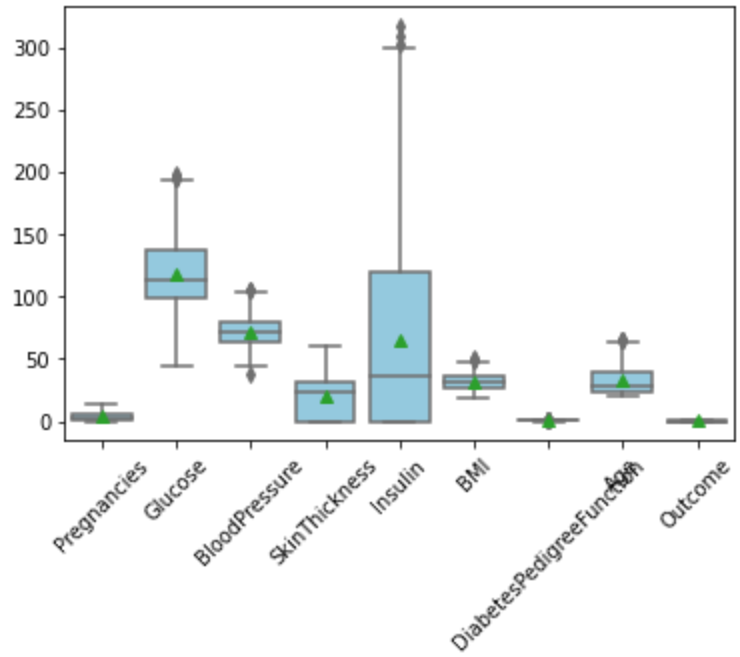
In [14]:
```
def outliers(data,feature):
    IQ1 = data[feature].quantile(0.25)
    IQ3 = data[feature].quantile(0.75)
    IQR = IQ3 - IQ1
    lower_bound = IQ1 - 1.5 * IQR
    upper_bound = IQ3 + 1.5 * IQR
    index = data.index[(data[feature] < lower_bound) | (data[feature] > upper_bc
    return index
```

In [15]:
```
index  = []
for i in data.columns:
    index.extend(outliers(data,i))
index = set(index)
print('total number of outliers are()'.format(len(index)))
data.drop(index,inplace = True,axis = 0)
data.shape
```

```
total number of outliers are()
```

```
(639, 9)
```

In [16]:
```
# Again Checking for the outliers
sns.boxplot(
    data=data,
    orient="v",
    showmeans=True,
    color='skyblue',
)
plt.xticks(rotation=45)
plt.show()
for column in data.columns:
    data[[column]].boxplot()
    plt.title(f'Boxplot for {column}')
    plt.ylabel('Values')
    plt.show()
```



Boxplot for Pregnancies

Most of the outliers are removed from our data We can now look for model selectic

Model selection,Training and determining Acuur

In [17]:
```
features = ['Glucose', 'BMI', 'Age','BloodPressure','Insulin']
```

Linear Regression

In [35]:
```
X_train, X_test, y_train, y_test = train_test_split(data[features], data['Outcom
# Training a logistic regression model to predict diabetes outcome
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)

# Evaluate the performance of the model on the test set
y_pred = lr_model.predict(X_test)
accuracy = np.mean(y_pred == y_test)

# Print the accuracy of the model
print('Accuracy of Linear Regressor Model :', accuracy)
```

```
Accuracy of Linear Regressor Model : 0.75625
```

Decision Tree

In [39]:
```
from sklearn.tree import DecisionTreeClassifier
# Train a decision tree model to predict diabetes outcome
dt_model = DecisionTreeClassifier()
dt_model.fit(X_train, y_train)

# Evaluate the performance of the model on the test set
y_pred_dt = dt_model.predict(X_test)
accuracy_dt = np.mean(y_pred_dt == y_test)

print('Accuracy of decision tree model:', accuracy_dt)
```

```
Accuracy of decision tree model: 0.6625
```

Random Forest Classifier

In [54]:
```
from sklearn.ensemble import RandomForestClassifier

# Train a random forest model to predict diabetes outcome
rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

# Evaluate the performance of the model on the test set
y_pred_rf = rf_model.predict(X_test)
accuracy_rf = np.mean(y_pred_rf == y_test)

print('Accuracy of random forest model:', accuracy_rf)
```

```
Accuracy of random forest model: 0.7875
```

The accuracies of Linear Regression, Decision Tree Classifier and Random Fores
74%,66%,78% respectively
So Random Forest Regressor Model outperforms both of them