In [1]:
```
# libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# loading the dataset
data=pd.read_csv("C:/Users/himanshi bajaj/OneDrive/Desktop/Diabetes_Dataset.csv")
data
```

Out[2]:

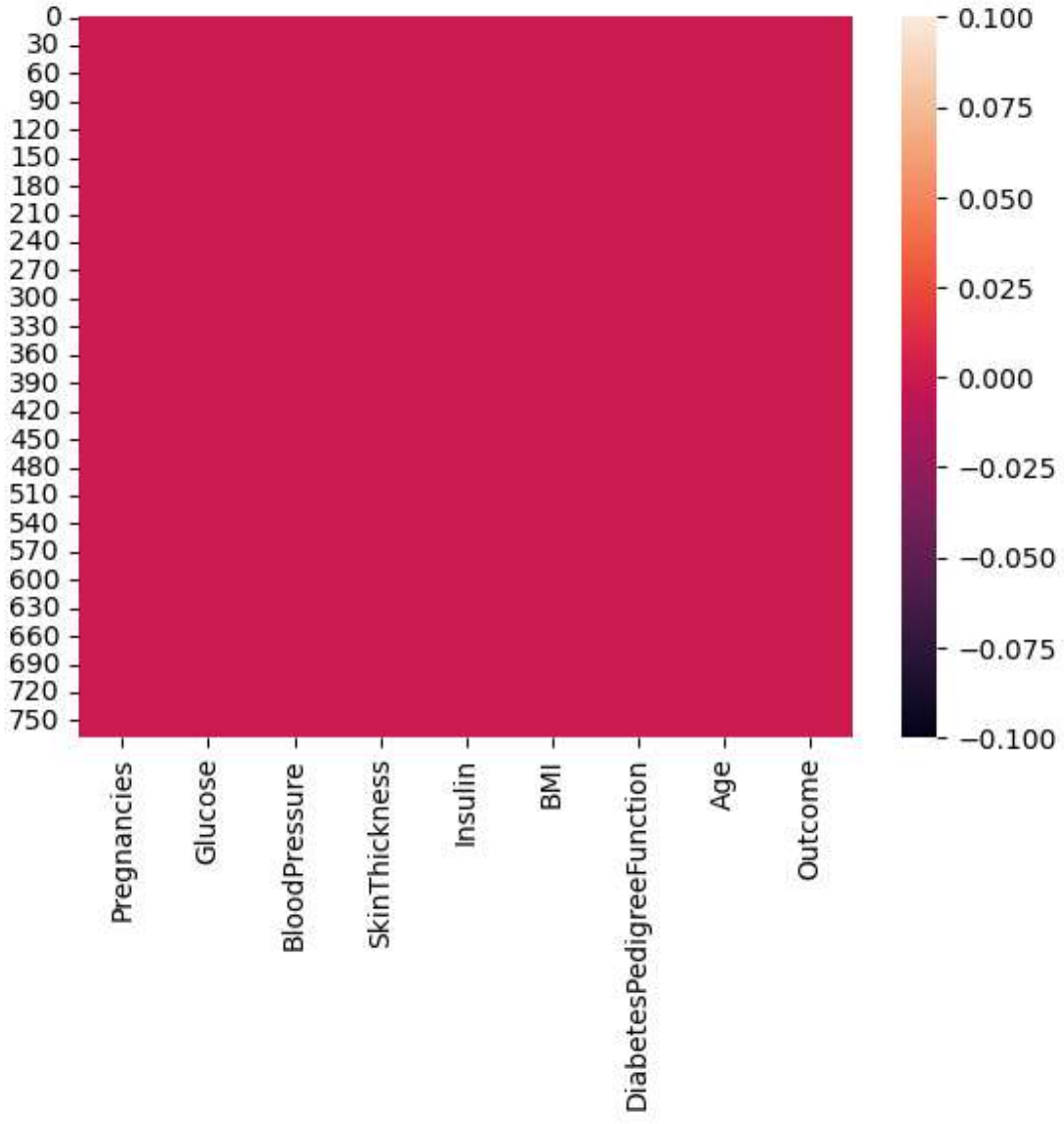
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	6	148	72	35	0	33.6	0.627	50
1	1	85	66	29	0	26.6	0.351	31
2	8	183	64	0	0	23.3	0.672	32
3	1	89	66	23	94	28.1	0.167	21
4	0	137	40	35	168	43.1	2.288	33
...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63
764	2	122	70	27	0	36.8	0.340	27
765	5	121	72	23	112	26.2	0.245	30
766	1	126	60	0	0	30.1	0.349	47
767	1	93	70	31	0	30.4	0.315	23

768 rows × 9 columns

In [3]:
```
# Checking for missing values
data.isnull().sum()
```

Out[3]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [5]:
```
# Checking for missing values
sns.heatmap(data.isnull())
plt.show()
```



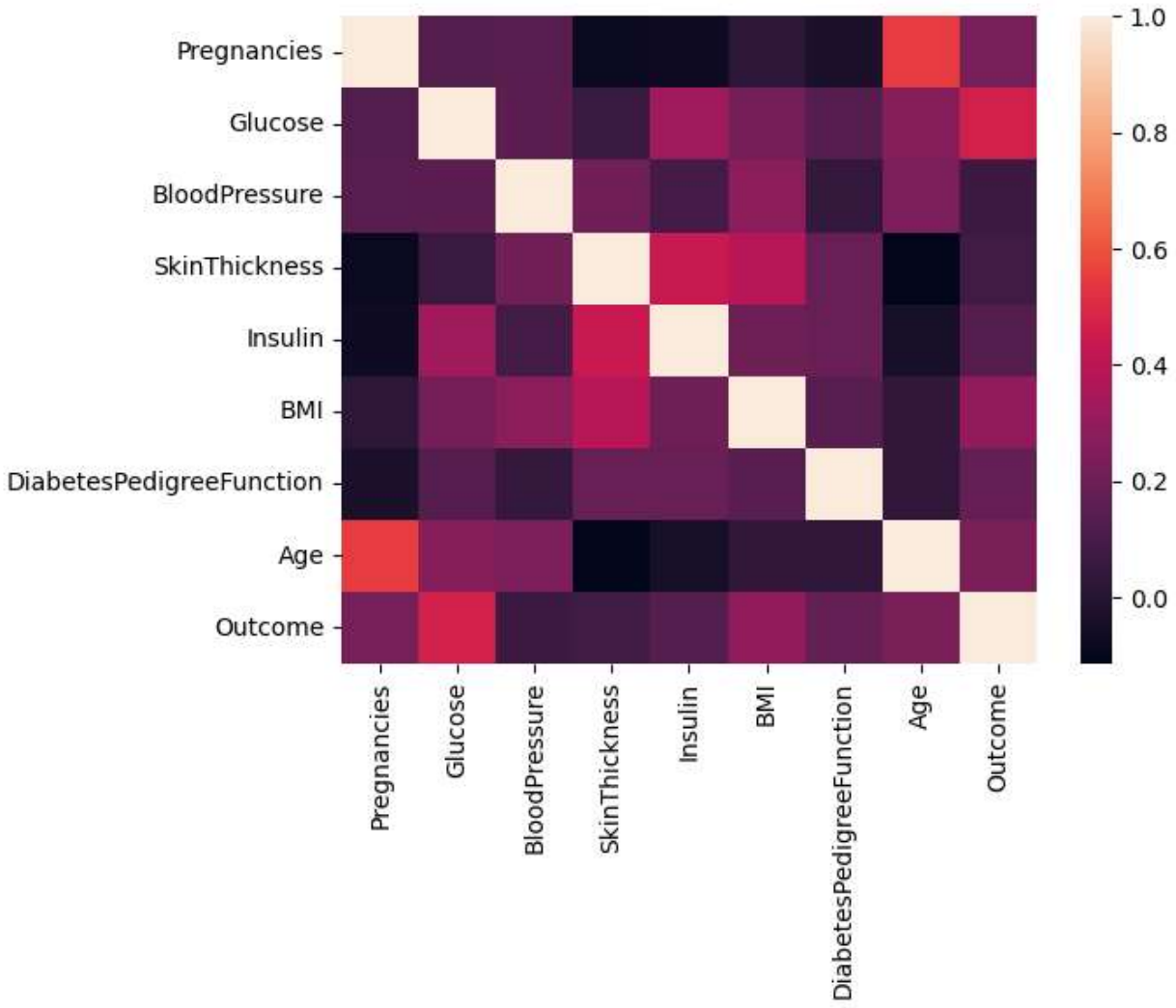
In [6]:
```
# Co relation matrix
correlation=data.corr()
print(correlation)
```

```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       1.000000       0.207371
SkinThickness               -0.081672  0.057328       0.207371       1.000000
Insulin                     -0.073535  0.331357       0.088933       0.436783
BMI                          0.017683  0.221071       0.281805       0.392573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183928
Age                          0.544341  0.263514       0.239528      -0.113970
Outcome                      0.221898  0.466581       0.065068       0.074752

                            Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies               -0.073535  0.017683                 -0.033523
Glucose                    0.331357  0.221071                  0.137337
BloodPressure              0.088933  0.281805                  0.041265
SkinThickness              0.436783  0.392573                  0.183928
Insulin                    1.000000  0.197859                  0.185071
BMI                        0.197859  1.000000                  0.140647
DiabetesPedigreeFunction   0.185071  0.140647                  1.000000
Age                       -0.042163  0.036242                  0.033561
Outcome                    0.130548  0.292695                  0.173844

                               Age   Outcome
Pregnancies               0.544341  0.221898
Glucose                   0.263514  0.466581
BloodPressure             0.239528  0.065068
SkinThickness            -0.113970  0.074752
Insulin                  -0.042163  0.130548
BMI                       0.036242  0.292695
DiabetesPedigreeFunction  0.033561  0.173844
Age                       1.000000  0.238356
Outcome                   0.238356  1.000000
```

In [8]:
```
# Visualising the Co relation
sns.heatmap(data.corr())
plt.show()
```



In [9]:
```
# Train test split
x=data.drop("Outcome",axis=1)
y=data["Outcome"]
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=101)
```

In [10]:
```
# Training the model
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
```

```
D:\Users\himanshi bajaj\anaconda3\Lib\site-packages\sklearn\linear_model\_logisti
c.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lea
rn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
(https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[10]:
```
LogisticRegression
LogisticRegression()
```

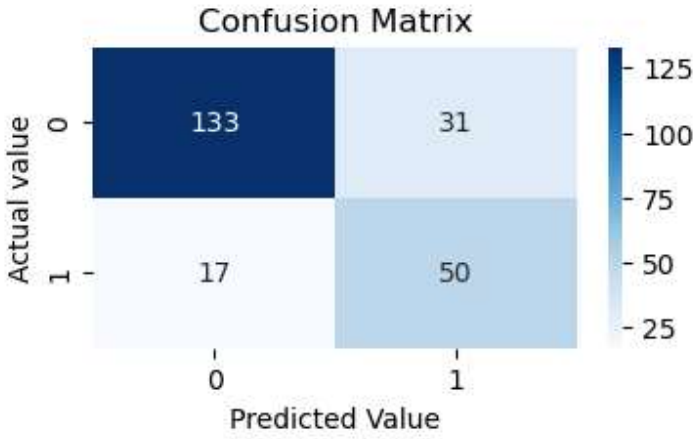
In [11]:
```
# Making Prediction
prediction = model.predict(x_test)
```

In [12]:
```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

In [13]:
```
cm=confusion_matrix(prediction,y_test)
cm
```

Out[13]:
```
array([[133,  31],
       [ 17,  50]], dtype=int64)
```

In [15]:
```
plt.figure(figsize=(4,2))
plt.title("Confusion Matrix")
sns.heatmap(cm,annot=True,fmt='d',cmap='Blues')
plt.ylabel("Actual value")
plt.xlabel("Predicted Value")
plt.show()
```



In [16]:
```
print(classification_report(prediction,y_test))
```

```
              precision    recall  f1-score   support

           0       0.89      0.81      0.85       164
           1       0.62      0.75      0.68        67

    accuracy                           0.79       231
   macro avg       0.75      0.78      0.76       231
weighted avg       0.81      0.79      0.80       231
```

In [17]:
```
accuracy = accuracy_score(prediction,y_test)
accuracy
```

Out[17]:
```
0.7922077922077922
```