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

In [2]:
```
#importing important libraries for prediction
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

#loading the dataset file = r.load("http://localhost:8888/edit/Diabetes/diabetes_dataset.csv" (http://localhost:8888/edit/Diabetes/diabetes_dataset.csv")) data = pd.read_csv(file)

In [3]:
```
#loading the dataset
data = pd.read_csv("diabetes_dataset.csv")
```

In [4]:
```
data
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedi
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	
...	...	...	...	...	...	...	
763	10	101	76	48	180	32.9	
764	2	122	70	27	0	36.8	
765	5	121	72	23	112	26.2	
766	1	126	60	0	0	30.1	
767	1	93	70	31	0	30.4	

768 rows × 9 columns

In [12]:
```
data.shape
```

Out[12]: (768, 9)

In [13]:
```
data.columns
```

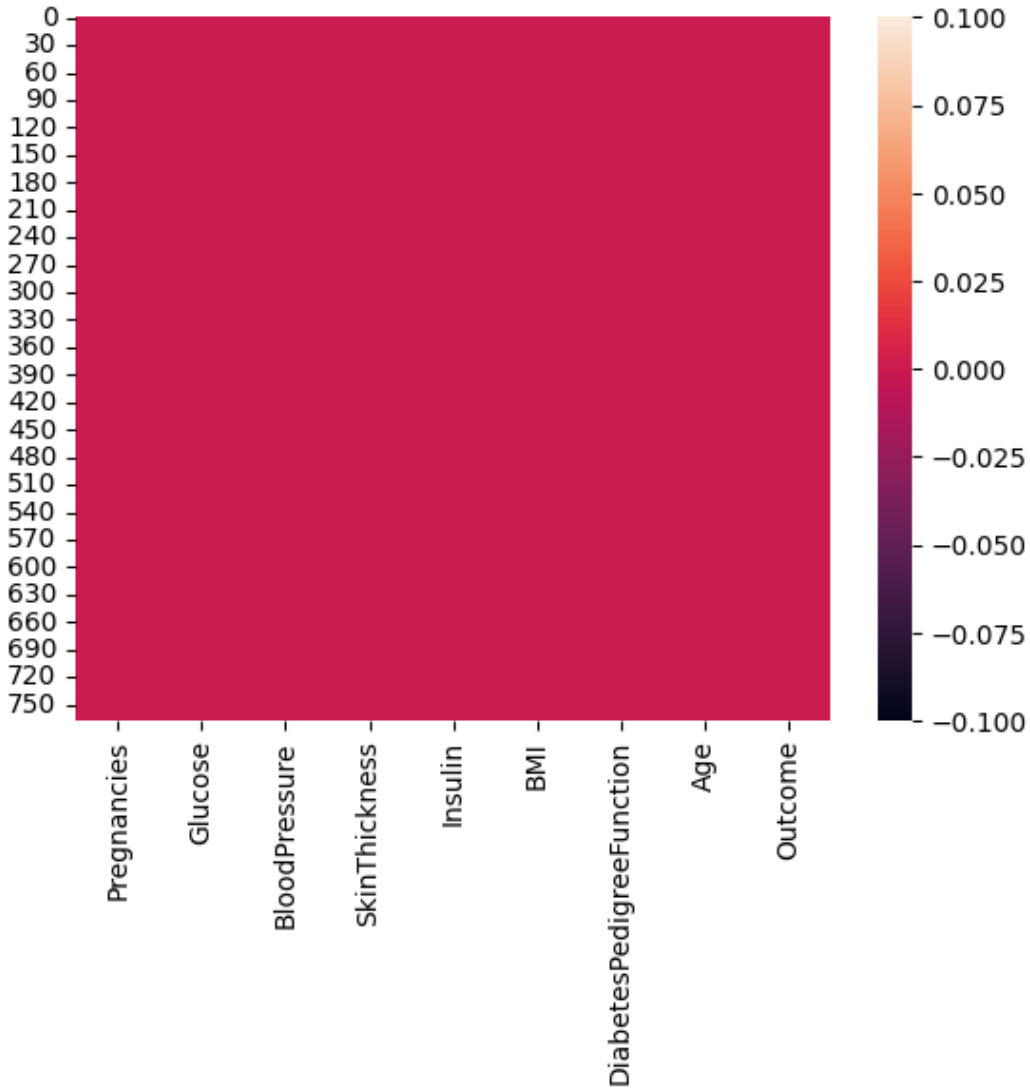
Out[13]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness',
'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [14]:
```
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 [5]:
```
#checking for missing values
sns.heatmap(data.isnull())
```

Out[5]: <Axes: >



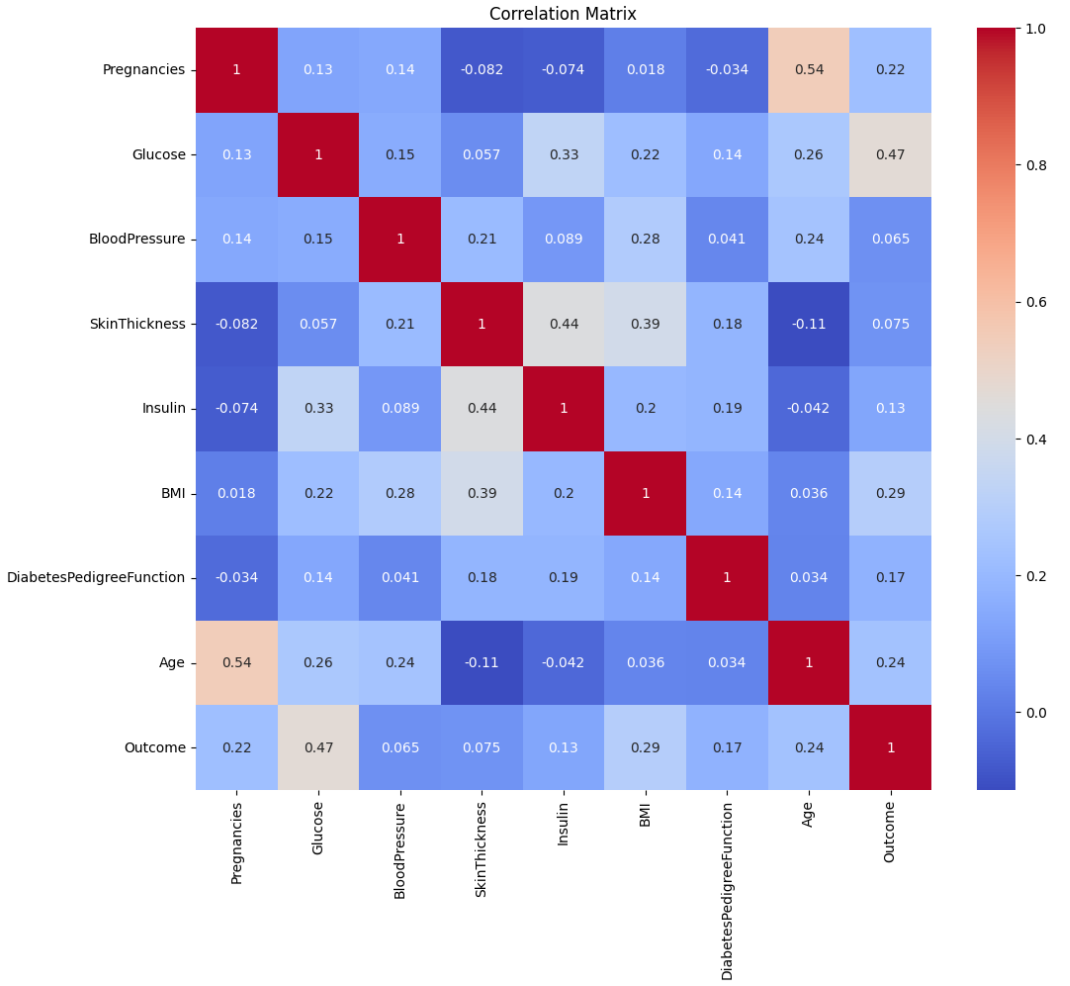
In [6]:
```
#Co relation matrix
correlation = data.corr()
print(correlation)
```

```
                          Pregnancies   Glucose  BloodPressure  Skin
Thickness  \
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  DiabetesPedigreeFuncti
on  \
Pregnancies              -0.073535  0.017683                -0.0335
23
Glucose                   0.331357  0.221071                 0.1373
37
BloodPressure             0.088933  0.281805                 0.0412
65
SkinThickness             0.436783  0.392573                 0.1839
28
Insulin                   1.000000  0.197859                 0.1850
71
BMI                       0.197859  1.000000                 0.1406
47
DiabetesPedigreeFunction  0.185071  0.140647                 1.0000
00
Age                      -0.042163  0.036242                 0.0335
61
Outcome                   0.130548  0.292695                 0.1738
44

                               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 [7]:
```
plt.figure(figsize=(12, 10))
p=sns.heatmap(correlation, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



In [8]:
```
X =data.drop("Outcome",axis=1)
Y =data['Outcome']
X_train,X_test,Y_train,Y_test =train_test_split(X,Y,test_size=0.2)
```

In [9]:
```
model = LogisticRegression(max_iter = 1000) #Adjusting the number of i
model.fit(X_train, Y_train)
```

Out[9]:
```
LogisticRegression
LogisticRegression(max_iter=1000)
```
(https://scikit-learn.org/1.4/modules/generated/sklearn.linear

In [10]:
```
#Making Prediction
predictions = model.predict(X_test)
print(predictions)
```

```
[1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 0
0 1 0
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1
0 0 0
 1 1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 1
0 0 0
 1 0 1 1 0 1 1 0 0 0 0 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 0 0 0 0 0 0 1 0
0 1 1
 0 1 0 0 0 1]
```

In [15]:
```
accuracy = accuracy_score(predictions,Y_test)
print(accuracy)
```

```
0.7987012987012987
```

This (accuracy) means that the model correctly predicted the outcome of diabetes in approximately 79.87% of the cases in the test set