Diabetes Prediction System

Importing Libraries & Dataset

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

import warnings
from warnings import catch_warnings
from warnings import filterwarnings
warnings.filterwarnings("ignore")
```

In [5]:
```
# Loading the dataset
data = pd.read_csv(r'C:\Users\priya\Downloads\diabetes.csv')
data
```

Out[5]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFu
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

Exploratory Data Abalysis (EDA)

In [6]:
```
# Exploring the dataset columns
data.columns
```

Out[6]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insul
in',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

In [7]:
```
# Getting information about the dataset
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 [8]:
```
# Getting dataset description
data.describe().T
```

Out[8]:

	count	mean	std	min	25%	50%	7
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00
Glucose	768.0	120.894531	31.972618	0.000	99.00000	117.0000	140.25
BloodPressure	768.0	69.105469	19.355807	0.000	62.00000	72.0000	80.00
SkinThickness	768.0	20.536458	15.952218	0.000	0.00000	23.0000	32.00
Insulin	768.0	79.799479	115.244002	0.000	0.00000	30.5000	127.25
BMI	768.0	31.992578	7.884160	0.000	27.30000	32.0000	36.60
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00

In [9]:
```
# Checking for messing values
data.isnull()
```

Out[9]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeF
0	False	False	False	False	False	False	
1	False	False	False	False	False	False	
2	False	False	False	False	False	False	
3	False	False	False	False	False	False	
4	False	False	False	False	False	False	
...	...	...	...	...	...	...	
763	False	False	False	False	False	False	
764	False	False	False	False	False	False	
765	False	False	False	False	False	False	
766	False	False	False	False	False	False	
767	False	False	False	False	False	False	

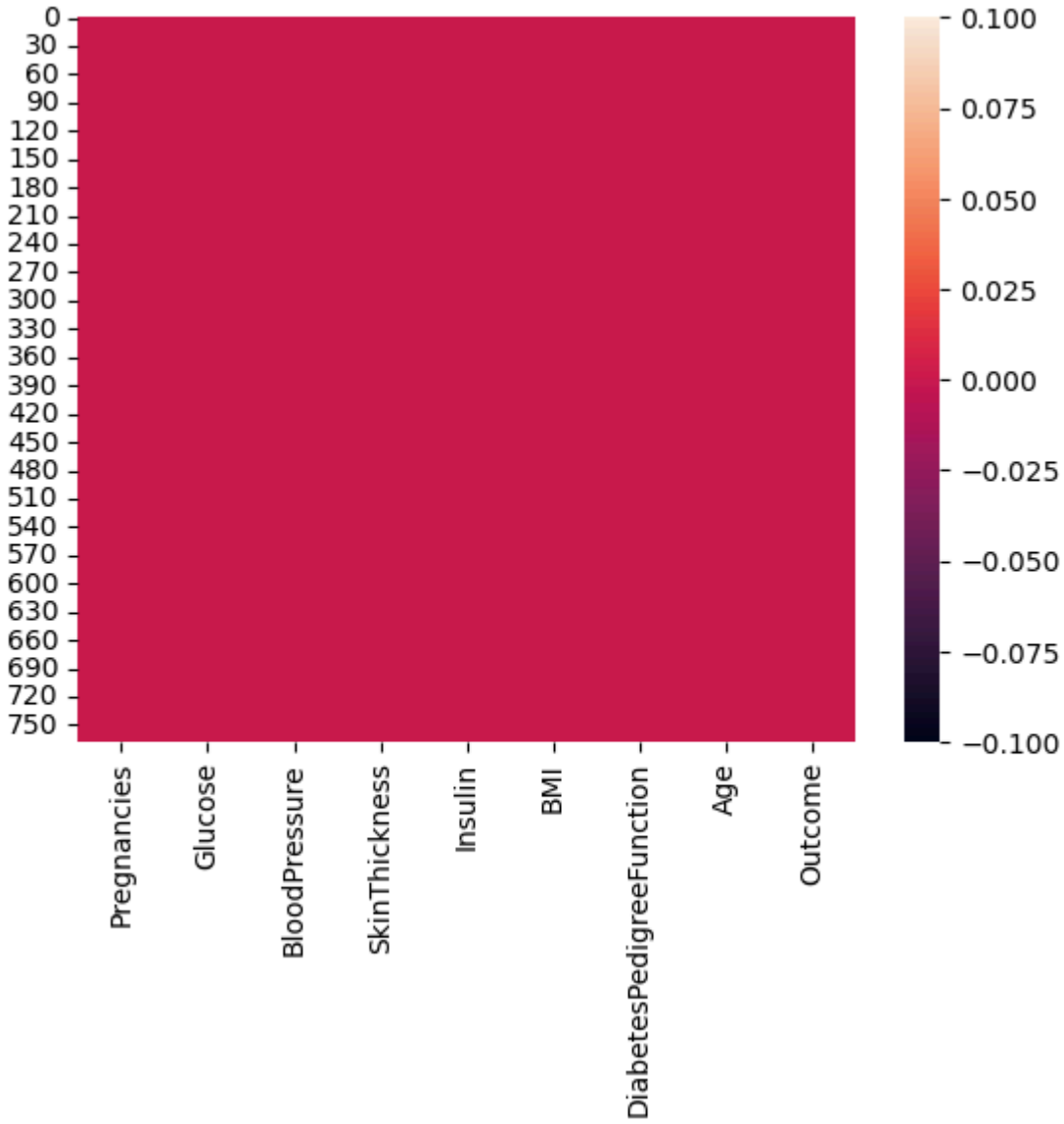
768 rows × 9 columns

In [10]:
```
# Checking the total of missing
data.isnull().sum()
```

Out[10]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

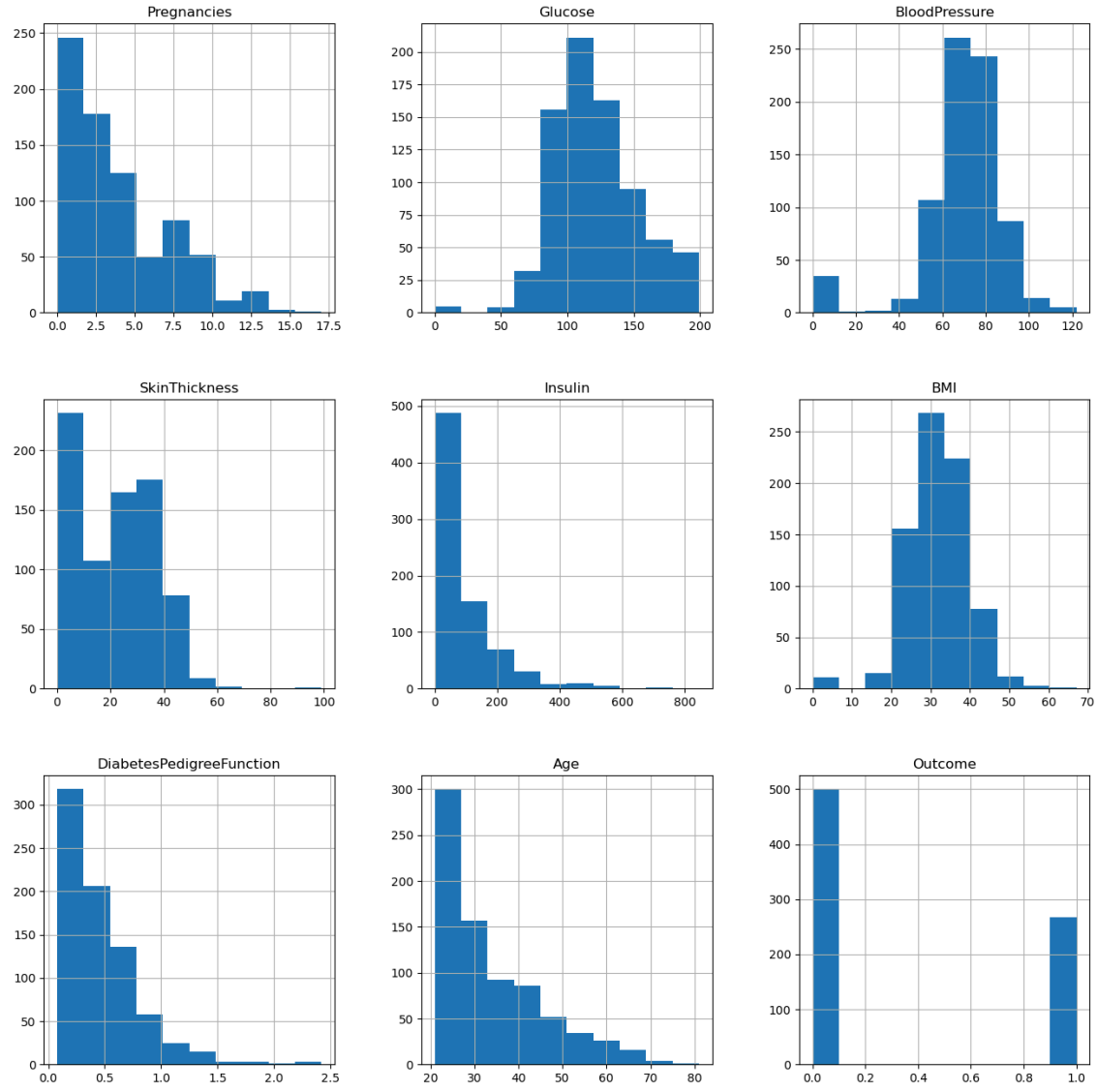
In [11]:
```
# Checking for missing values as heatmap
sns.heatmap(data.isnull())
```

Out[11]: <Axes: >



Data Visualisation

In [12]:
```
# Plotting the data distribution plots
data.hist(figsize=(16,16))
plt.show()
```



In [13]:
```
# Plotting data pair plots
sns.pairplot(data, hue='Outcome')
plt.show()
```

Correlation Heatmap

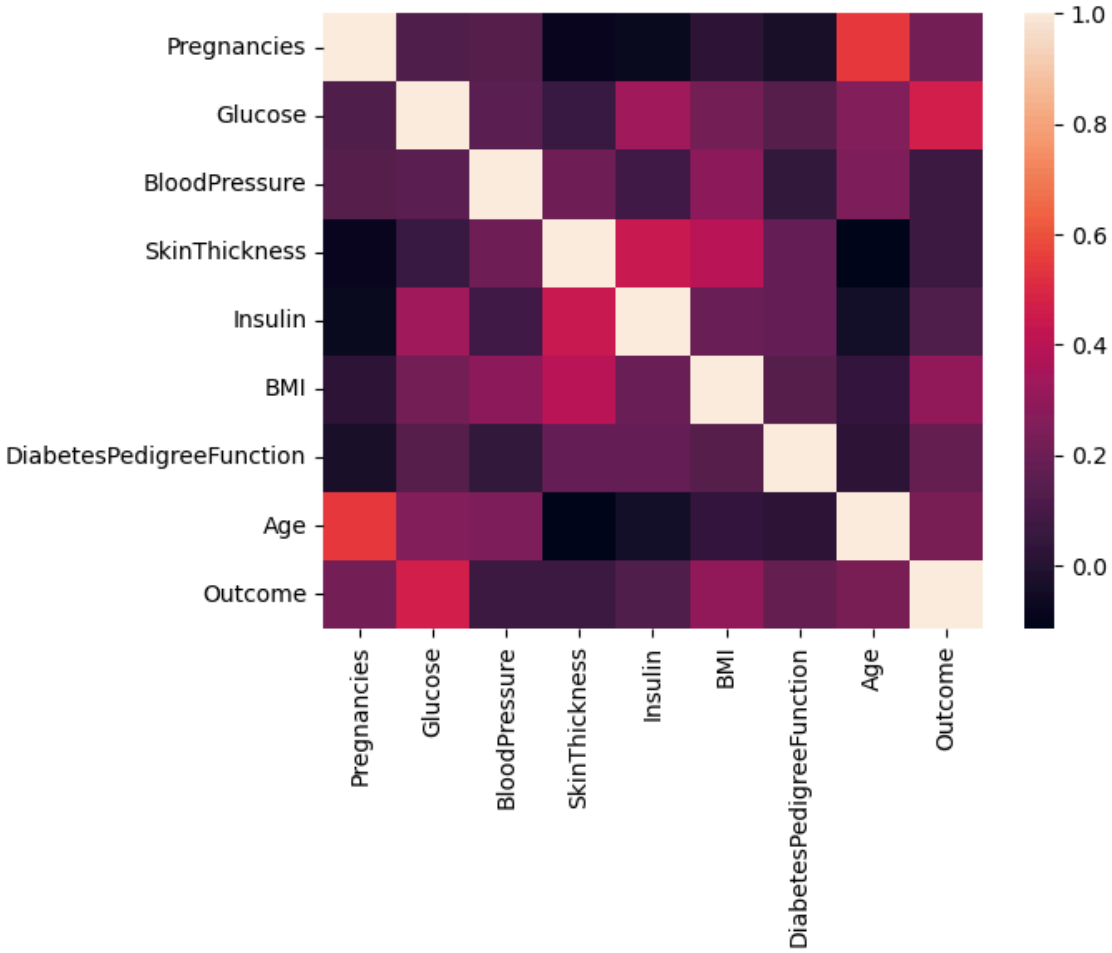
In [14]:
```
# Calculating the correlation matrix
correlation = data.corr()
correlation
```

Out[14]:

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

In [15]:
```
# Visualising the correlation heatmap
sns.heatmap(correlation)
```

Out[15]: <Axes: >



Training the Model

In [16]:
```
# Spliting the data
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 [17]:
```
# Training the model
model = LogisticRegression()
model.fit(X_train,Y_train)
```

Out[17]:
```
▾ LogisticRegression
LogisticRegression()
```

The Prediction

In [18]:
```
# Making the prediction
prediction = model.predict(X_test)
print(prediction)
```

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

In [19]:
```
# Calculating the accuracy
accuracy = accuracy_score(prediction, Y_test)
print(accuracy)
```

```
0.8051948051948052
```

Conclusion

- The model has a precision of 80%
- Patients with high Blood Pressure has greater chances of diabetes.

End!