### JELILAT OLUWATOSIN ABDULLATEEF

Data Analyst

abdullateefjelilat25@gmail.com (mailto:abdullateefjelilat25@gmail.com) /
www.linkedin.com/in/jelilat (http://www.linkedin.com/in/jelilat)


# DIABETES PATIENT PREDICTION ANALYSIS

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether a patient has diabetes based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger dataset. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

# Importing Libraries

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

#### *Importing libraries for prediction*

In [2]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
```

#### *loading the Dataset*

In [3]:
```
df = pd.read_csv("diabetes.csv")
```

# Exploratory Data Analysis (EDA)

#### First five rows of the Dataset

In [4]:
```
df.head()
```

Out[4]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunction |
|---|---|---|---|---|---|---|---|
| **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 |

### Last five rows of the dataset

In [5]:
```
df.tail()
```

Out[5]:

| | Pregnancies | Glucose | BloodPressure | SkinThickness | Insulin | BMI | DiabetesPedigreeFunctio |
|---|---|---|---|---|---|---|---|
| **763** | 10 | 101 | 76 | 48 | 180 | 32.9 | 0.17 |
| **764** | 2 | 122 | 70 | 27 | 0 | 36.8 | 0.34 |
| **765** | 5 | 121 | 72 | 23 | 112 | 26.2 | 0.24 |
| **766** | 1 | 126 | 60 | 0 | 0 | 30.1 | 0.34 |
| **767** | 1 | 93 | 70 | 31 | 0 | 30.4 | 0.31 |

### Overview of the dataset

In [6]:
```
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
```

### Shape of the dataset

In [7]:
```
df.shape
```

Out[7]:
```
(768, 9)
```

**Available Column names**

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

**Data type in each column**

In [9]:
```
df.dtypes
```

Out[9]:
```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

# Checking for duplicate data

In [10]:
```
df.duplicated().sum()
```

Out[10]:
```
0
```

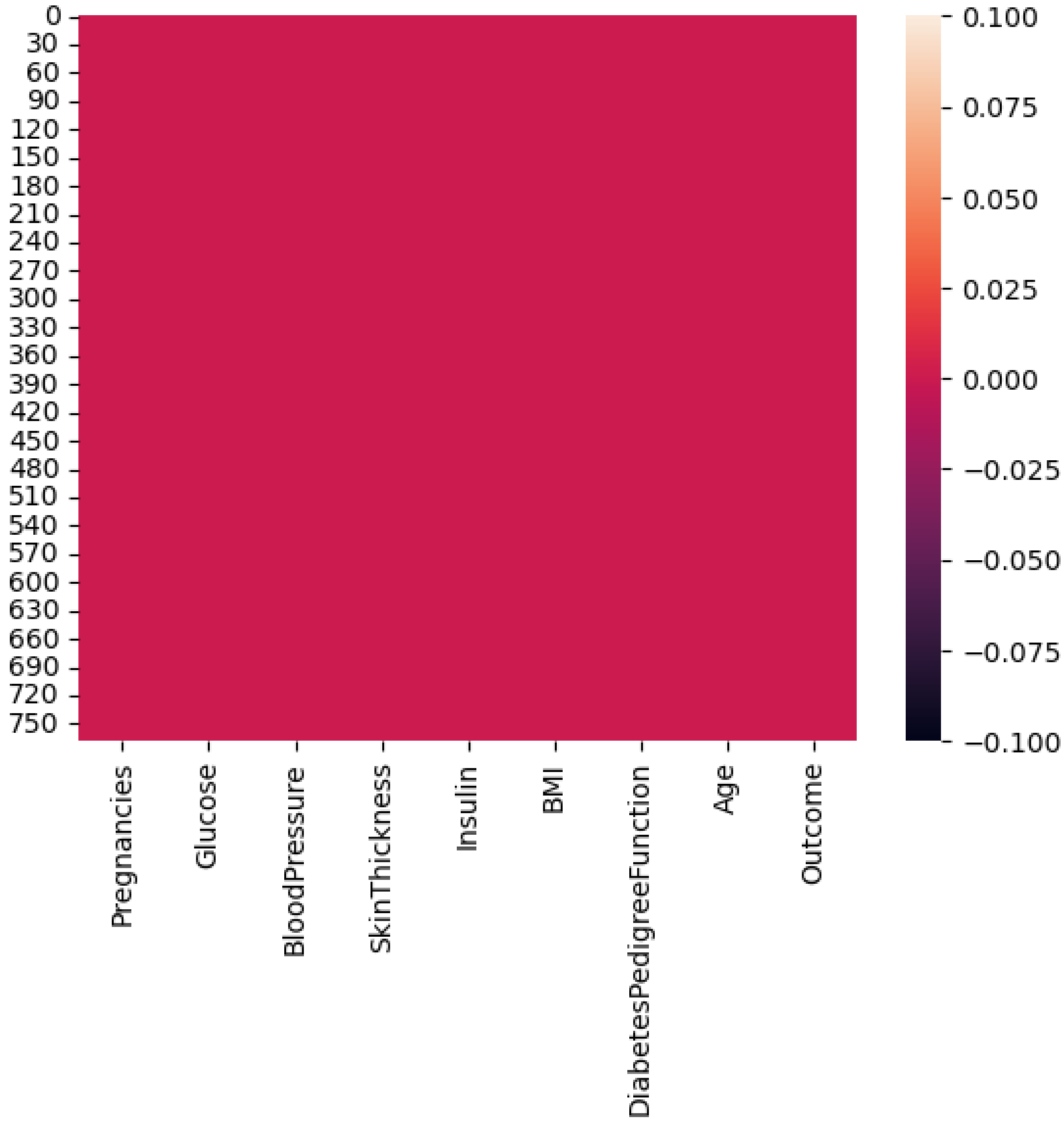
# Checking for missing values

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [12]:
```
sns.heatmap(df.isnull())
```

Out[12]: <Axes: >



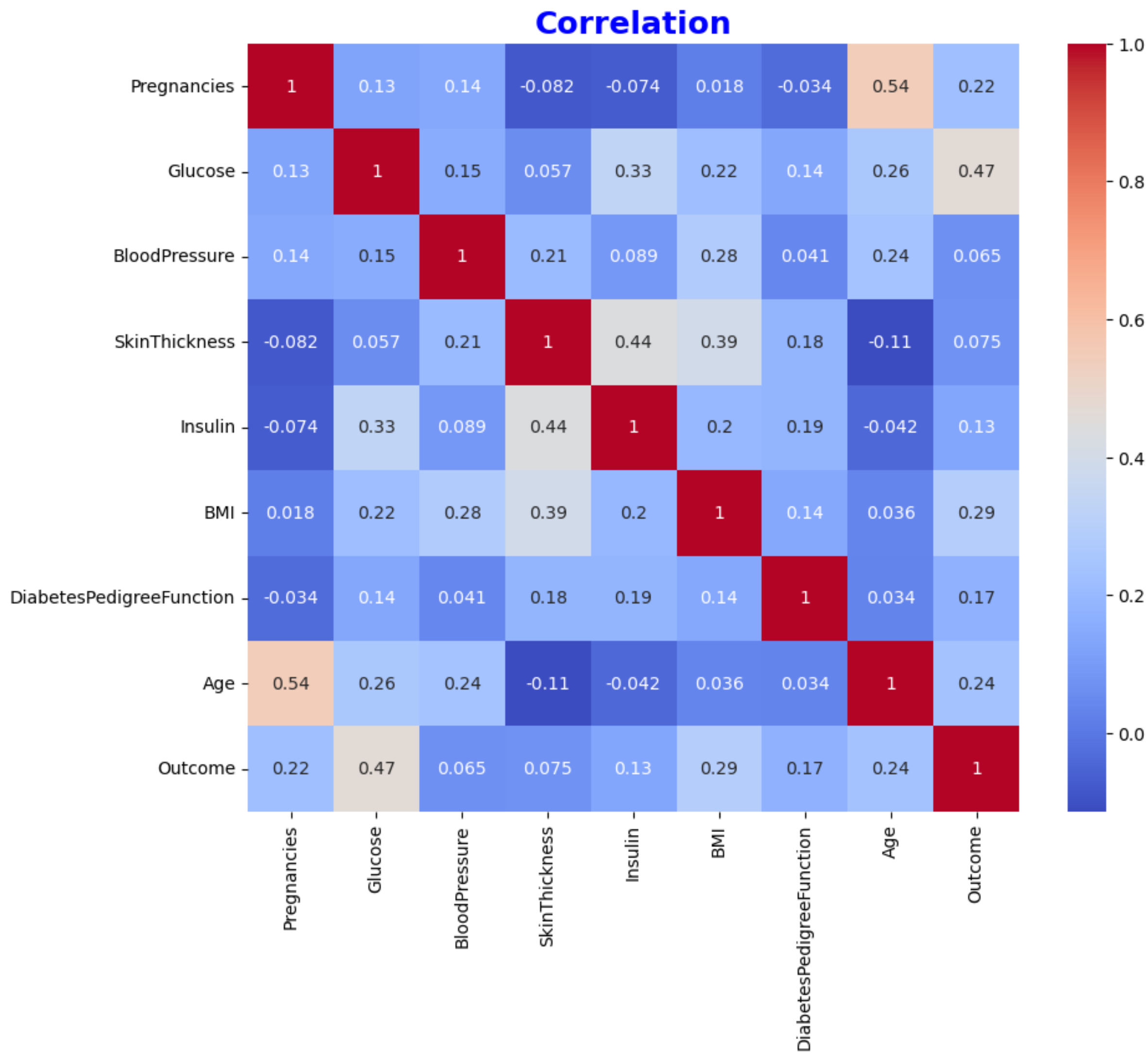

# Correlation Matrix

In [13]:
```
correlation=df.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 [14]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True, cmap='coolwarm')
plt.title('Correlation', color = 'blue',fontweight='bold',fontsize=18)

plt.show
```

Out[14]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```




# Training the Model with Train Test Split

**Train test split**

Train-test split is a techniques used in machine learning to assess model performance. It divides the dataset into a training set and a testing set, with a 0.2 test size indicating that 20% of the data is used for testing and 80% for training.

In [15]:
```
x=df.drop("Outcome",axis=1)
y=df['Outcome']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_s
```

In X all the imdependent variables are stored In Y the predictor variable ("Outcome") is stored.

In [16]:
```
scaler = StandardScaler()
x_train_scaler = scaler.fit_transform(x_train)
x_test_scaler = scaler.transform(x_test)

print(x_test_scaler.shape,x_train_scaler.shape)
```

```
(154, 8) (614, 8)
```

**Training the Model**

Fitting the x train and y train data into the variable called model.

In [17]:
```
model=LogisticRegression()
model.fit(x_train_scaler,y_train)
```

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

# Making Prediction

In [18]:
```
prediction = model.predict(x_test_scaler)
print(prediction)
```

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

After training the model, predictions are made using the data, which comprises 20% of the total datasets.

In [19]:
```
accuracy = accuracy_score(prediction,y_test)
print('Accuracy:-',accuracy)
```

```
Accuracy:- 0.7532467532467533
```

## ACCURACY:- 75.32%

The model predicted the presence or absence of diabetes in approximately 75.32% of the cases