DIABETES PATIENT PREDICTION ANALYSIS

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

importing libraries

In [17]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing libraries for prediction

In [53]:
```
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 [5]:
```
import pandas as pd
```

In [6]:
```
Dia = pd.read_csv("diabetes.csv")
Dia
```

Out[6]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	C
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

First five rows of the Dataset

In [7]:
```
Dia.head()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Out
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	

Last five rows of the dataset

In [8]:
```
Dia.tail()
```

Out[8]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	C
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	

Dataset info

In [9]:
```
Dia.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
```

Columns names

In [10]:
```
Dia.columns
```

Out[10]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

Data types

In [11]:
```
Dia.dtypes
```

Out[11]:
```
Pregnancies                 int64
Glucose                     int64
BloodPressure               int64
SkinThickness               int64
Insulin                     int64
BMI                       float64
DiabetesPedigreeFunction  float64
Age                         int64
Outcome                     int64
dtype: object
```

checking for replicated data

In [13]:
```
Dia.duplicated().sum()
```

Out[13]:
```
0
```

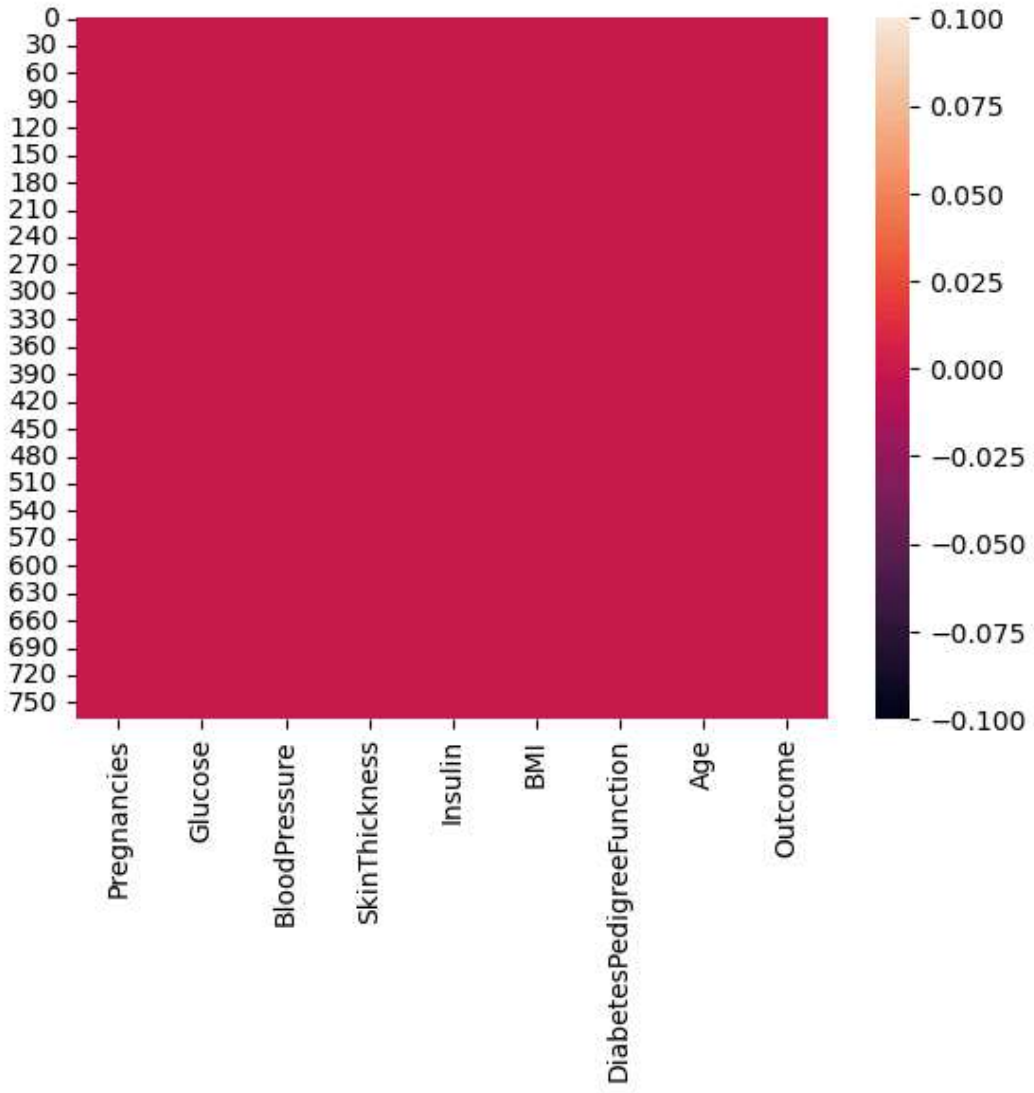
checking for Errors

In [12]:
```
Dia.isnull().sum()
```

Out[12]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [18]:
```
sns.heatmap(Dia.isnull())
```

Out[18]: <Axes: >



Correlation Matrix

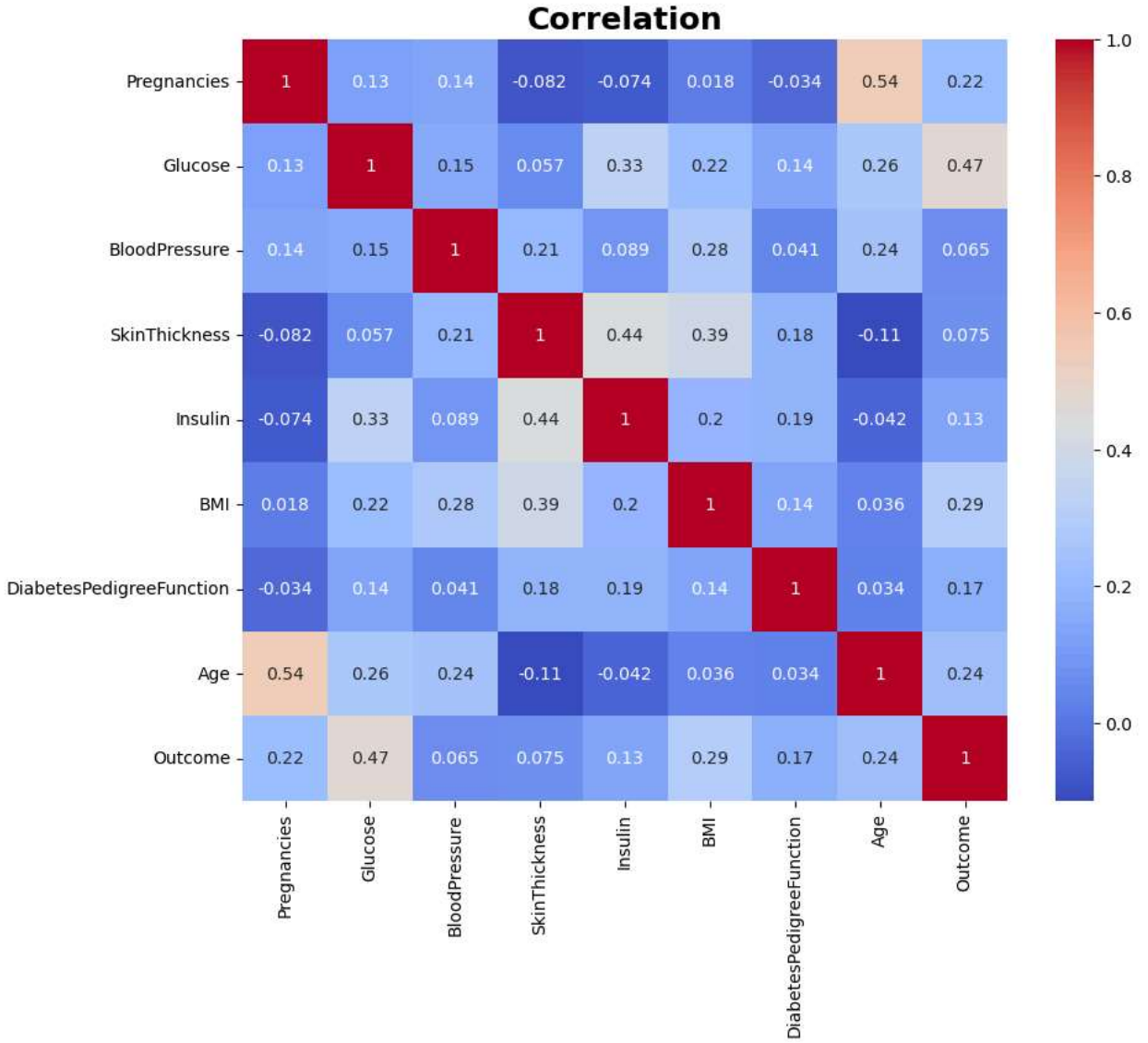
In [22]:
```
correlation=Dia.corr()
print(correlation)
```

```
                          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 [34]:
```
plt.figure(figsize=(10, 8))
sns.heatmap(Dia.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation', color='black', fontweight='bold', fontsize=18)  # Corrected sy
plt.show()
```



Training the Model with Train Test Split

Train test split

Train 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 [44]:
```
x=Dia.drop("Outcome",axis=1)
y=Dia['Outcome']
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2, random_state=42)
```

In [58]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_train_scaler = scaler.fit_transform(x_train)
x_test_scaler = scaler.transform(x_test)
print(x_train_scaler.shape, x_test_scaler.shape)
```

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

Training the model

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

In [59]:
```
model=LogisticRegression()
model.fit(x_train_scaler,y_train)
```

Out[59]:
```
LogisticRegression
LogisticRegression()
```

Making Prediction

In [66]:
```
prediction = model.predict(x_test_scaler)
```

In [67]:
```
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 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
 0 1 0 0 0 0]
```

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

In [71]:
```
accuracy = accuracy_score(prediction,y_test)
```

In [72]:
```
print('Accuracy:-',accuracy)
```

```
Accuracy:- 0.7532467532467533
```

ACCURACY :- 75.32%

The model predicted the presence or absence of diabeties in approximately 75% of the cases.

In [ ]: