DIABETES PREDICTION ANALYSIS

IMPORTING PACKAGES

In [13]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
```

DATA COLLECTION

In [2]:

```
df=pd.read_csv('diabetes-dataset.csv', encoding='unicode_escape')
```

In [3]:

```
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	2	138	62	35	0	33.6	0.127	47	1
1	0	84	82	31	125	38.2	0.233	23	0
2	0	145	0	0	0	44.2	0.630	31	1
3	0	135	68	42	250	42.3	0.365	24	1
4	1	139	62	41	480	40.7	0.536	21	0

In [4]:

```
df.shape
```

Out[4]:

```
(2000, 9)
```

DATA CLEANING

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               2000 non-null   int64  
 1   Glucose                   2000 non-null   int64  
 2   BloodPressure             2000 non-null   int64  
 3   SkinThickness             2000 non-null   int64  
 4   Insulin                   2000 non-null   int64  
 5   BMI                       2000 non-null   float64
 6   DiabetesPedigreeFunction  2000 non-null   float64
 7   Age                       2000 non-null   int64  
 8   Outcome                   2000 non-null   int64  
dtypes: float64(2), int64(7)
memory usage: 140.8 KB
```

In [6]:

```
pd.isnull(df).sum()
```

Out[6]:

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [7]:

```
df.describe()
```

Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	3.703500	121.182500	69.145500	20.935000	80.254000	32.193000	0.470930	33.090500
std	3.306063	32.068636	19.188315	16.103243	111.180534	8.149901	0.323553	11.786423
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000
25%	1.000000	99.000000	63.500000	0.000000	0.000000	27.375000	0.244000	24.000000
50%	3.000000	117.000000	72.000000	23.000000	40.000000	32.300000	0.376000	29.000000
75%	6.000000	141.000000	80.000000	32.000000	130.000000	36.800000	0.624000	40.000000
max	17.000000	199.000000	122.000000	110.000000	744.000000	80.600000	2.420000	81.000000

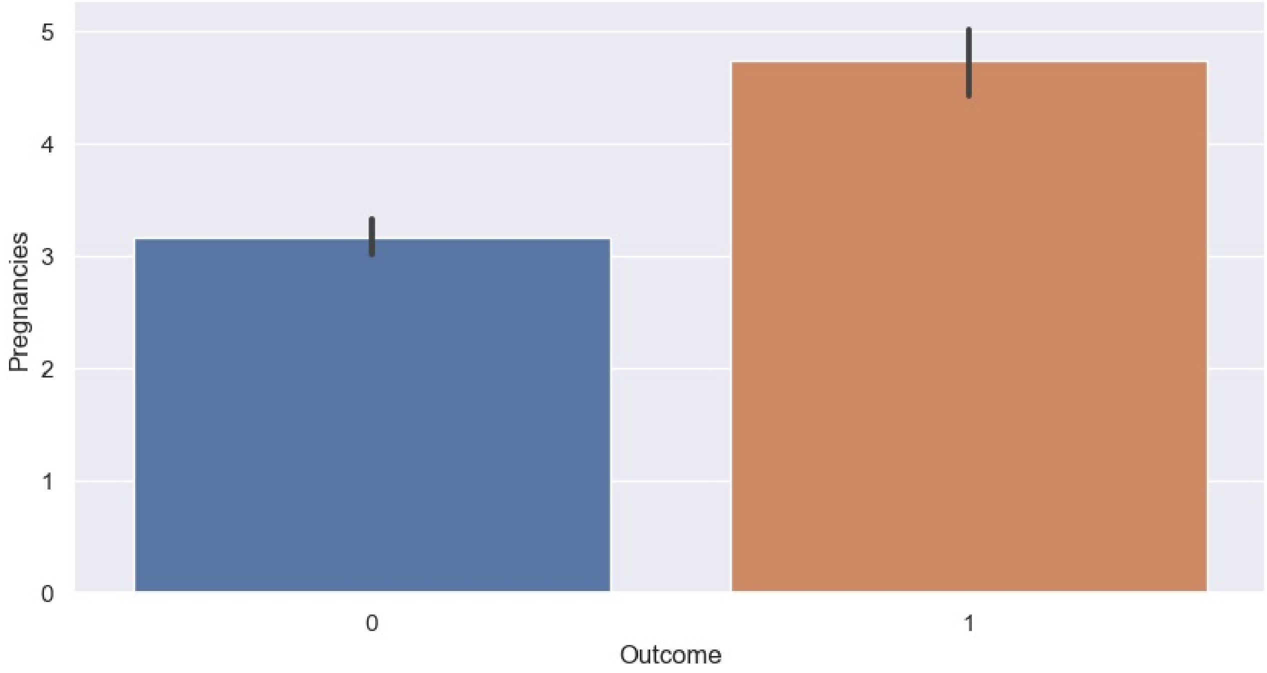
DATA VISUALIZATION

In [12]:

```
sns.set(rc={'figure.figsize':(10,5)})
sns.barplot(x='Outcome',y='Pregnancies',data=df)
```

Out[12]:

```
<Axes: xlabel='Outcome', ylabel='Pregnancies'>
```



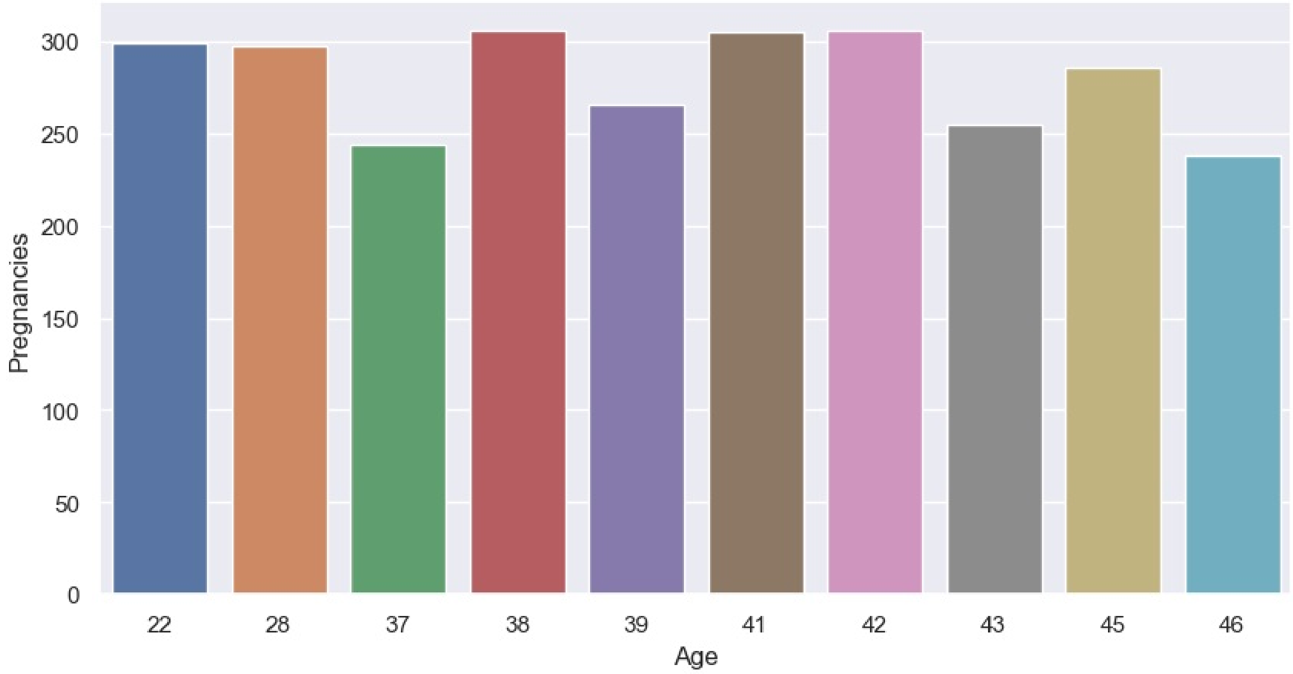
In [46]:

```
diabetes_df=df.groupby(['Age'],as_index=False)['Pregnancies'].sum().sort_values(by='Pregnancies',ascending=Fals

sns.barplot(x='Age',y='Pregnancies',data=diabetes_df)
```

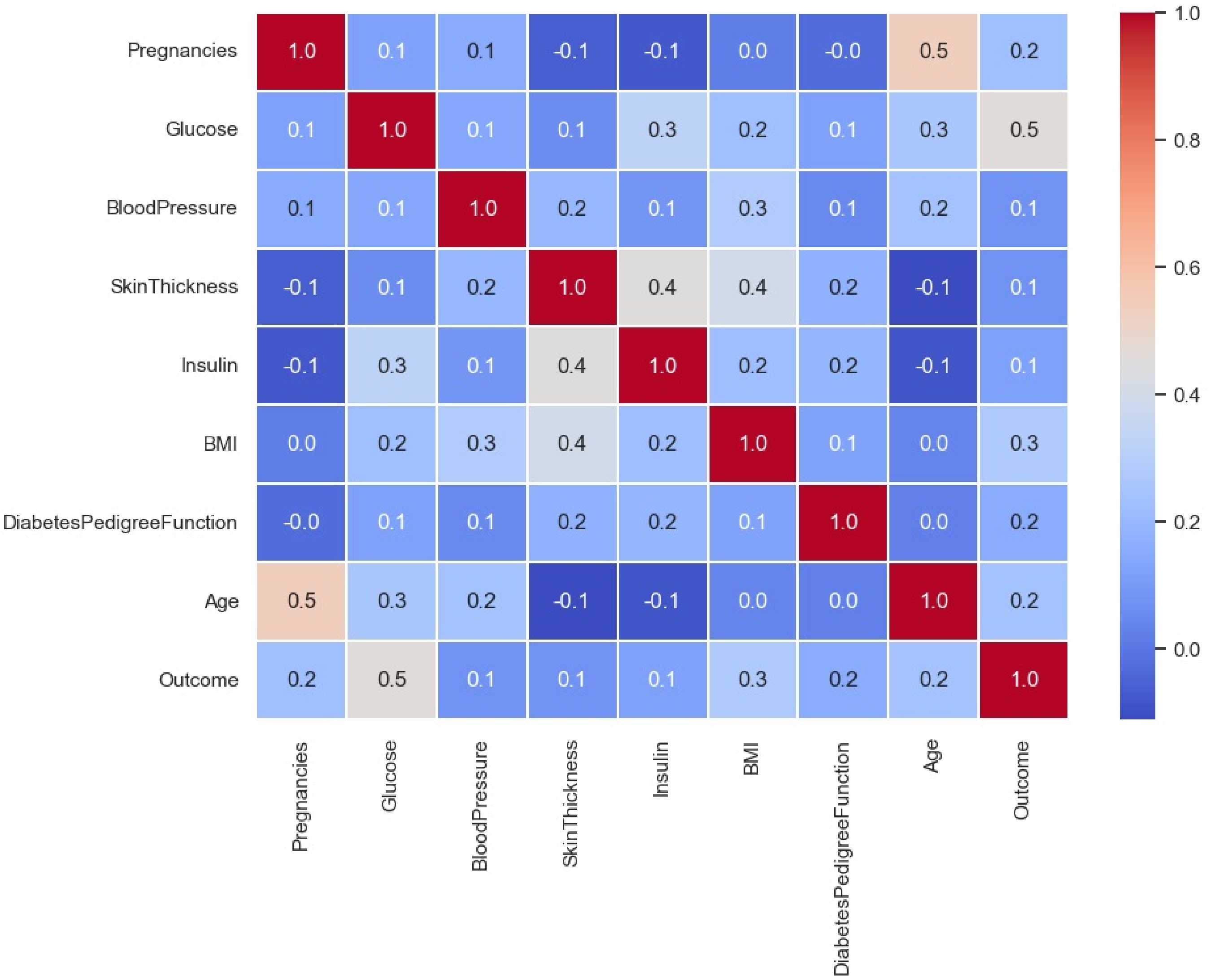
Out[46]:

```
<Axes: xlabel='Age', ylabel='Pregnancies'>
```



```
In [16]: plt.figure(figsize=(10, 7))

         sns.heatmap(df.corr(), annot=True, linewidths=0.2, fmt='.1f', cmap='coolwarm')
         plt.show()
```



FEATURE SELECTION

In [27]:

```
feature_columns =df[['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age']]

feature_columns.head()
```

Out[27]:

	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age
0	138	62	35	0	33.6	0.127	47
1	84	82	31	125	38.2	0.233	23
2	145	0	0	0	44.2	0.630	31
3	135	68	42	250	42.3	0.365	24
4	139	62	41	480	40.7	0.536	21

In [28]:

```
outcome_column =df['Outcome']
outcome_column.head()
```

Out[28]:

```
0    1
1    0
2    1
3    1
4    0
Name: Outcome, dtype: int64
```

In [29]:

```
from sklearn.model_selection import train_test_split
```

In [30]:

```
X_train, X_test, y_train, y_test = train_test_split( feature_columns, outcome_column, test_size=0.2, random_sta
```

In [31]:

```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(1600, 7)
(400, 7)
(1600,)
(400,)
```

LOADING MODEL FOR PREDICITION

```
In [32]: from sklearn.linear_model import LogisticRegression
         from sklearn.metrics import accuracy_score, confusion_matrix
```

```
In [33]: model = LogisticRegression()
```

```
In [34]: model = model.fit(X_train, y_train)
```

```
In [35]: score = model.predict(X_train)
```

MODEL TESTING AND EVALUATION

```
In [36]: print("Training Score: ", model.score(X_train, y_train))
         print("Testing Score:  ", model.score(X_test, y_test))
```

```
Training Score:  0.76375
Testing Score:   0.7825
```

```
In [37]: pred = model.predict(X_test)
         print("Model Accuracy is : ", pred)
```

```
Model Accuracy is :  [0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0
 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1 0 1 1 0 0 0 0 0
 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0
 1 0 0 1 1 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 1
 0 0 0 1 0 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 1 0 0 0 0 0
 0 1 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0
 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 0
 1 0 1 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 1 0 0 1 0 1 1 0 0 0 1 1 0 1
 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1 1 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 0 0 0 0
 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1]
```

```
In [38]: model.intercept_
```

```
Out[38]: array([-7.93210317])
```

```
In [39]: model.coef_
```

```
Out[39]: array([[ 0.03281421, -0.00826399,  0.00221131, -0.00136201,  0.07054979,
                  0.76528042,  0.0341431 ]])
```

```
In [40]: accuracy_score(y_test, pred)
```

```
Out[40]: 0.7825
```

```
In [41]: df.columns
```

```
Out[41]: Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
                'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
               dtype='object')
```

THANK YOU!

CONNECT WITH ME:

LinkedIn: https://www.linkedin.com/in/harshita-sharma-b68154220/

GitHub: https://github.com/DATAPREDICTS

Instagram: https://www.instagram.com/datapredicts?utm_source=qr&igsh=czVzc2k5c3oxOWQ4

YouTube: https://youtube.com/@Datapredicts?si=eDKAqVciVxg23zab

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js