Importing Libraries

In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
df = pd.read_csv('kaggle_diabetes.csv')
```

In [3]:

```
df.head()
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunct
0	2	138	62	35	0	33.6	0.
1	0	84	82	31	125	38.2	0.
2	0	145	0	0	0	44.2	0.
3	0	135	68	42	250	42.3	0.
4	1	139	62	41	480	40.7	0.

In [4]:

```
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 [5]:

```
df.describe().T
```

Out[5]:

	count	mean	std	min	25%	50%	75%	
Pregnancies	2000.0	3.70350	3.306063	0.000	1.000	3.000	6.000	1
Glucose	2000.0	121.18250	32.068636	0.000	99.000	117.000	141.000	19
BloodPressure	2000.0	69.14550	19.188315	0.000	63.500	72.000	80.000	12
SkinThickness	2000.0	20.93500	16.103243	0.000	0.000	23.000	32.000	11
Insulin	2000.0	80.25400	111.180534	0.000	0.000	40.000	130.000	74
BMI	2000.0	32.19300	8.149901	0.000	27.375	32.300	36.800	8
DiabetesPedigreeFunction	2000.0	0.47093	0.323553	0.078	0.244	0.376	0.624	
Age	2000.0	33.09050	11.786423	21.000	24.000	29.000	40.000	8
Outcome	2000.0	0.34200	0.474498	0.000	0.000	0.000	1.000	

In [6]:

```
df.isnull().any()
```

Out[6]:

```
Pregnancies                 False
Glucose                     False
BloodPressure               False
SkinThickness               False
Insulin                     False
BMI                         False
DiabetesPedigreeFunction    False
Age                         False
Outcome                     False
dtype: bool
```

In [7]:

```
df = df.rename(columns={'DiabetesPedigreeFunction':'DPF'})
df.head()
```

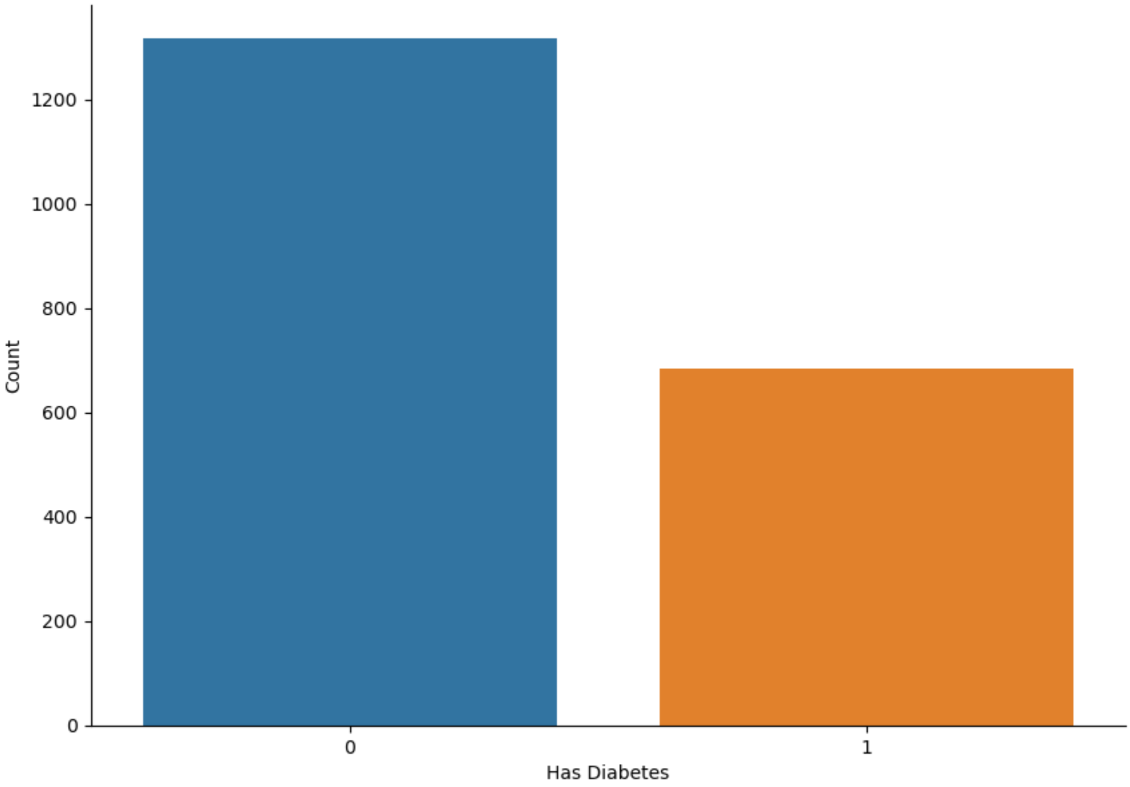
Out[7]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DPF	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 [8]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

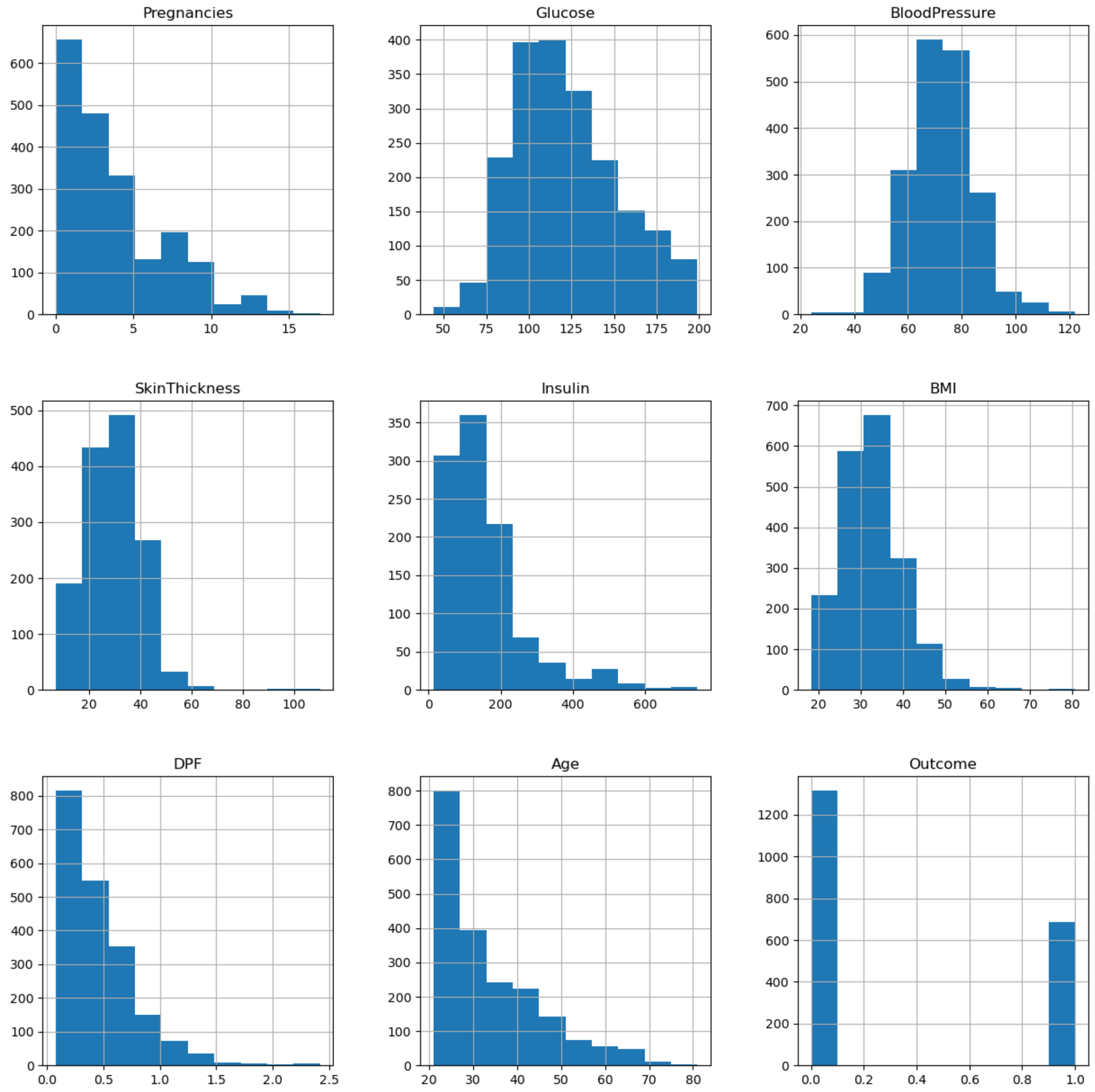
In [9]:
```
plt.figure(figsize=(10,7))
sns.countplot(x='Outcome', data=df)

# Removing the unwanted spines
plt.gca().spines['top'].set_visible(False)
plt.gca().spines['right'].set_visible(False)

# Headings
plt.xlabel('Has Diabetes')
plt.ylabel('Count')

plt.show()
```



In [10]:
```
df_copy = df.copy(deep=True)
df_copy[['Glucose','BloodPressure','SkinThickness','Insulin','BMI']] = df_c
df_copy.isnull().sum()
```

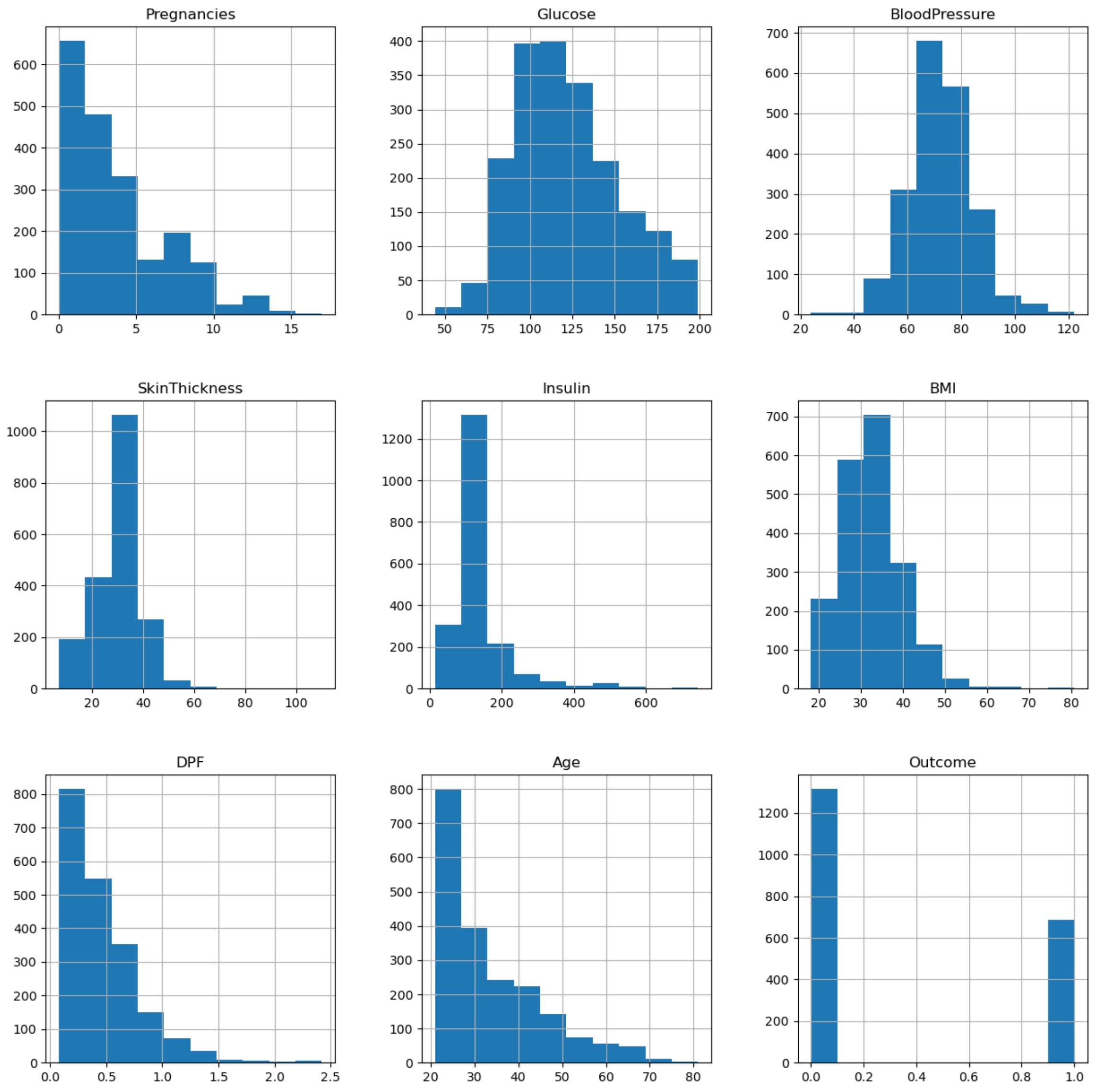
Out[10]:
```
Pregnancies         0
Glucose            13
BloodPressure      90
SkinThickness     573
Insulin           956
BMI                28
DPF                 0
Age                 0
Outcome             0
dtype: int64
```

In [11]:
```
p = df_copy.hist(figsize = (15,15))
```



In [12]:
```
df_copy['Glucose'].fillna(df_copy['Glucose'].mean(), inplace=True)
df_copy['BloodPressure'].fillna(df_copy['BloodPressure'].mean(), inplace=Tr
df_copy['SkinThickness'].fillna(df_copy['SkinThickness'].median(), inplace=
df_copy['Insulin'].fillna(df_copy['Insulin'].median(), inplace=True)
df_copy['BMI'].fillna(df_copy['BMI'].median(), inplace=True)
```

In [13]:
```
p = df_copy.hist(figsize=(15,15))
```



Modelling

In [14]:
```
from sklearn.model_selection import train_test_split

X = df.drop(columns='Outcome')
y = df['Outcome']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, r
print('X_train size: {}, X_test size: {}'.format(X_train.shape, X_test.shap
```

```
X_train size: (1600, 8), X_test size: (400, 8)
```

In [15]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [16]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import ShuffleSplit
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

In [17]:

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(RandomForestClassifier(n_estimators=20, random_sta
print('Average Accuracy : {}%'.format(round(sum(scores)*100/len(scores)), 3
```

```
Average Accuracy : 95%
```