Appendix

In [39]:
```
!jupyter nbconvert --to webpdf --allow-chromium-download Coursework.ipynb
```

```
[NbConvertApp] Converting notebook Coursework.ipynb to webpdf
[NbConvertApp] Building PDF
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 929033 bytes to Coursework.pdf
```

In [2]:
```
pip install pandas seaborn
```

```
Requirement already satisfied: pandas in c:\users\reeth\anaconda3\lib\site-packages (1.
5.3)Note: you may need to restart the kernel to use updated packages.

Requirement already satisfied: seaborn in c:\users\reeth\anaconda3\lib\site-packages (0.
12.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\reeth\anaconda3\lib\si
te-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\reeth\anaconda3\lib\site-package
s (from pandas) (2022.7)
Requirement already satisfied: numpy>=1.21.0 in c:\users\reeth\anaconda3\lib\site-packag
es (from pandas) (1.24.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in c:\users\reeth\anaconda3\lib\s
ite-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\reeth\anaconda3\lib\site-pac
kages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.0.5)
Requirement already satisfied: cycler>=0.10 in c:\users\reeth\anaconda3\lib\site-package
s (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\reeth\anaconda3\lib\site-pa
ckages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\reeth\anaconda3\lib\site-pa
ckages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\users\reeth\anaconda3\lib\site-pack
ages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\reeth\anaconda3\lib\site-packag
es (from matplotlib!=3.6.1,>=3.1->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\reeth\anaconda3\lib\site-pac
kages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.0.9)
Requirement already satisfied: six>=1.5 in c:\users\reeth\anaconda3\lib\site-packages (f
rom python-dateutil>=2.8.1->pandas) (1.16.0)
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.distributions.empirical_distribution import ECDF
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
from sklearn.metrics import mean_squared_error
import statsmodels.api as sm
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
```

In [4]:
```
#Reading PimaDiabetes.csv file into df dataframe
df=pd.read_csv('PimaDiabetes.csv')
df.head()
```

Out[4]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1

1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

EDA using Visualization

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

Out[5]:

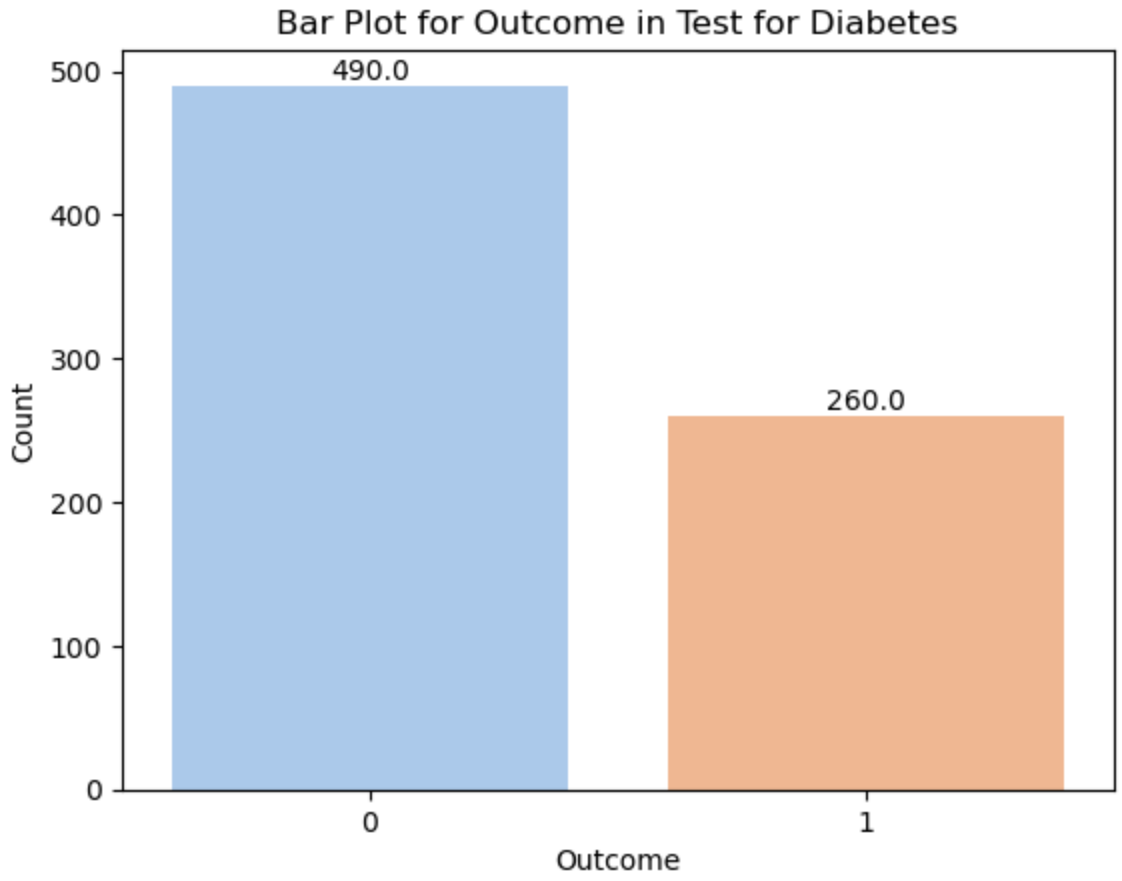
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Ag
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.0000
mean	3.844000	120.737333	68.982667	20.489333	80.378667	31.959067	0.473544	33.1666
std	3.370085	32.019671	19.508814	15.918828	115.019198	7.927399	0.332119	11.7088
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.0000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.244000	24.0000
50%	3.000000	117.000000	72.000000	23.000000	36.500000	32.000000	0.377000	29.0000
75%	6.000000	140.750000	80.000000	32.000000	129.750000	36.575000	0.628500	40.7500
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.0000

In [6]:
```
print("No of rows with value 0 in the following columns")
print("\nGlucose:")
print(len(df[df['Glucose']==0]))
print("\nBlood Pressure:")
print(len(df[df['BloodPressure']==0]))
print("\nSkin Thickness:")
print(len(df[df['SkinThickness']==0]))
print("\nInsulin:")
print(len(df[df['Insulin']==0]))
print("\nBMI:")
print(len(df[df['BMI']==0]))
```

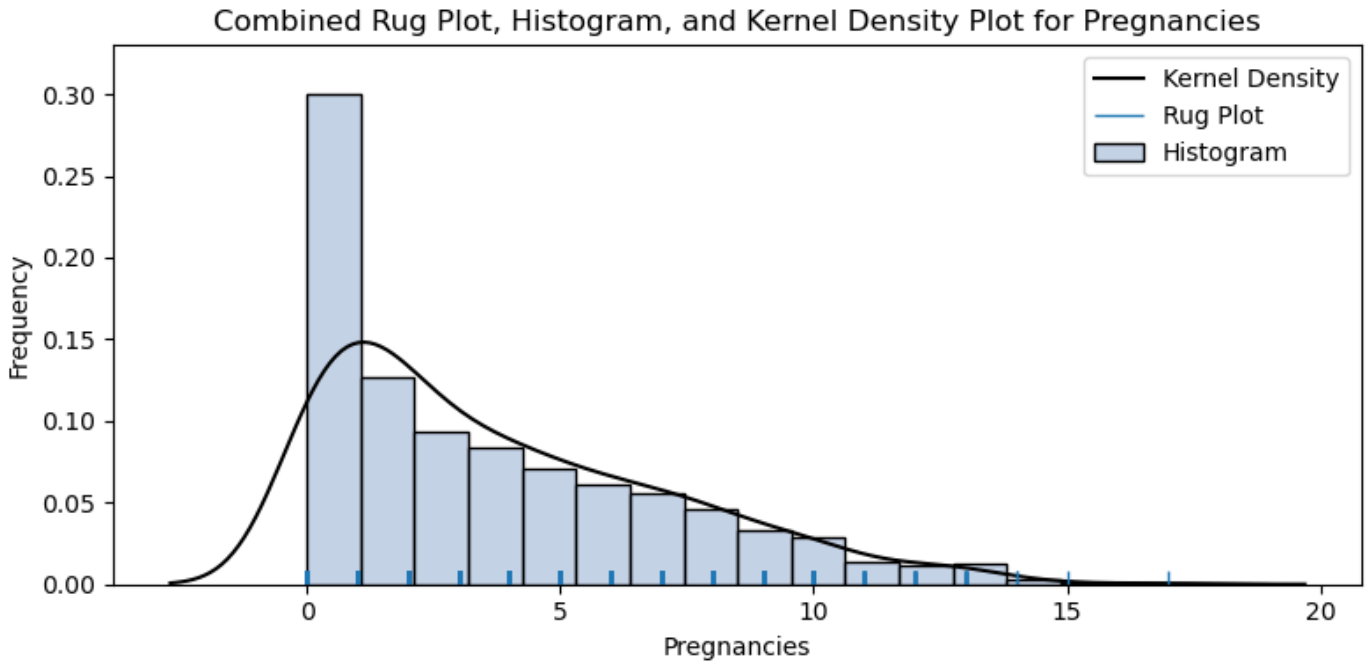
```
No of rows with value 0 in the following columns

Glucose:
5

Blood Pressure:
35

Skin Thickness:
221

Insulin:
362

BMI:
11
```

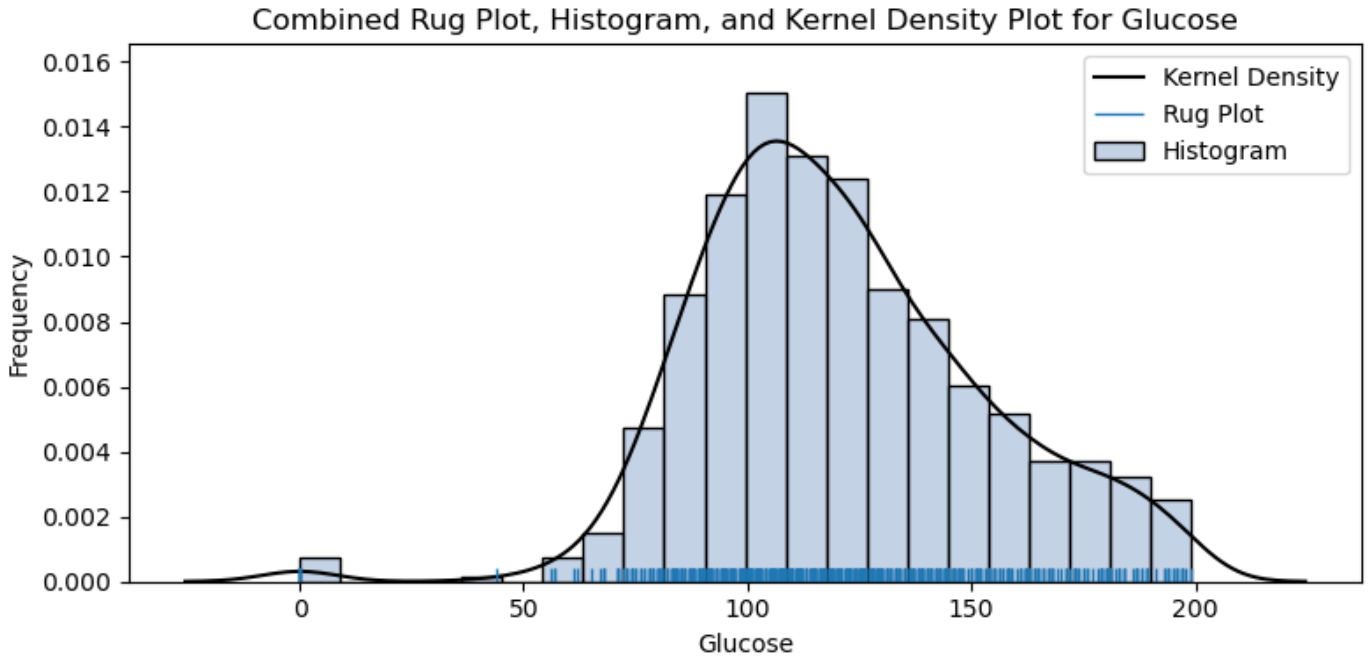
In [7]:
```
#Plotting Bar Plot to know how many women in the dataset have diabetes
bp=sns.countplot(x='Outcome', data=df, palette='pastel')
for i in bp.patches:
    x = i.get_x() + i.get_width() / 2
    y = i.get_height()
```

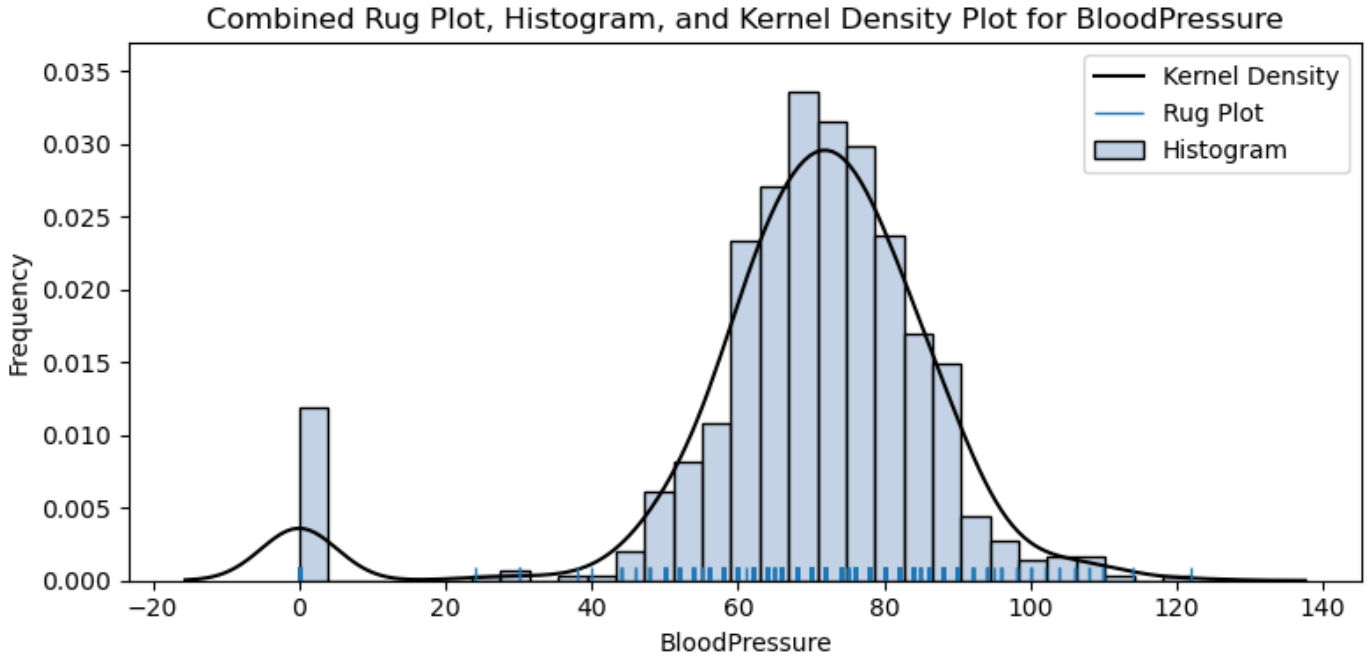
```
    bp.annotate(y, (x, y), ha='center', va='bottom')
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.title('Bar Plot for Outcome in Test for Diabetes')
plt.show()
```

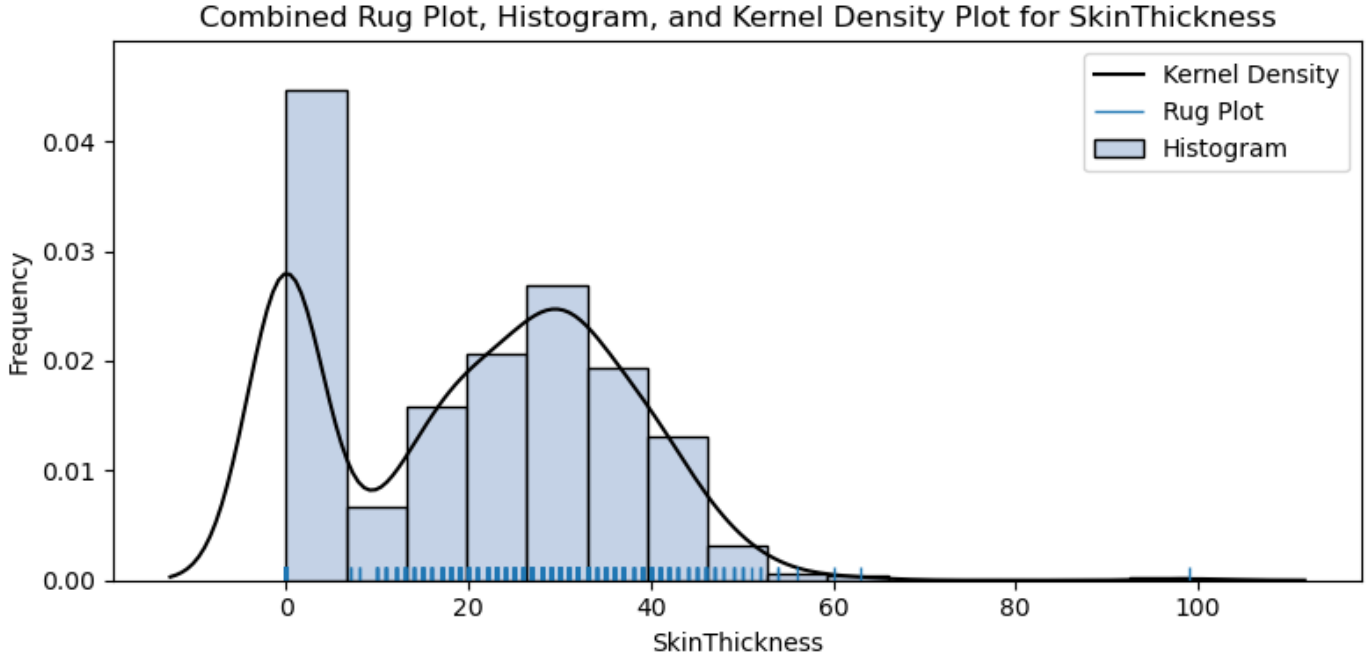


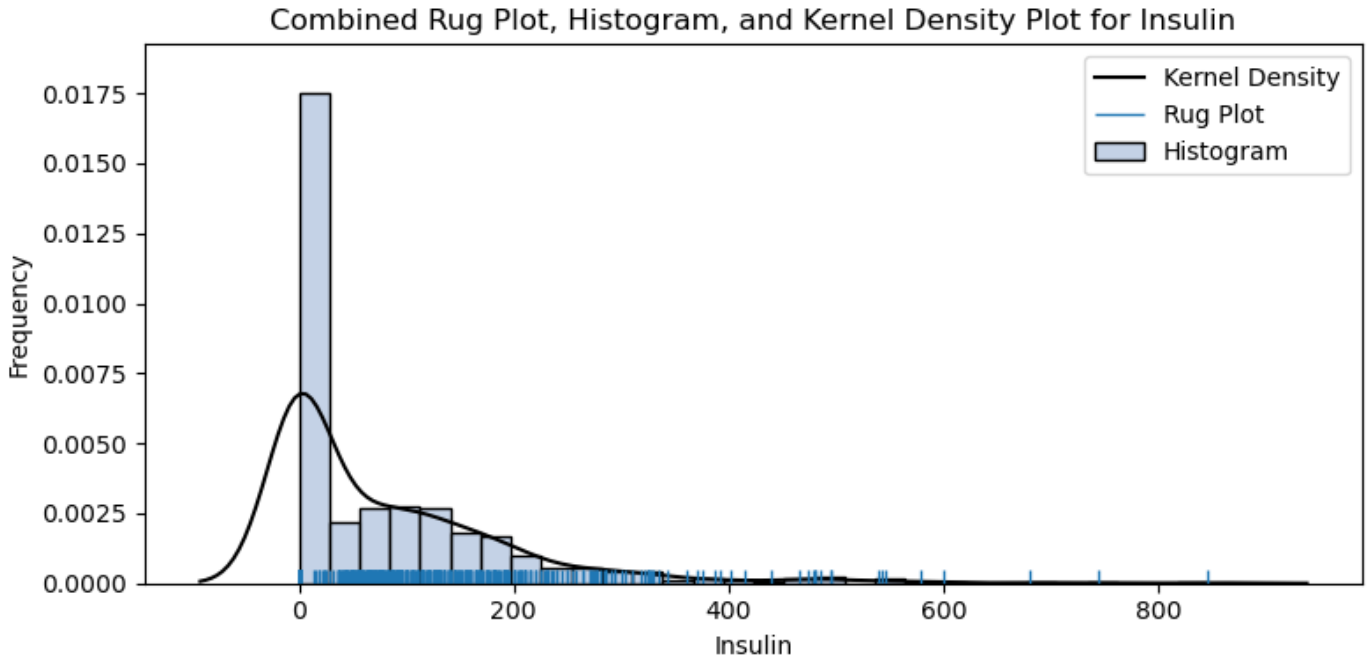
In [8]:
```
#PLotting a combined Rug Plot, Histogram and Kernel Density Plot for each column in the
for col in ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Dia

    plt.figure(figsize=(8, 4))
    seaborn_version_str = sns.__version__
    version_str_parts = seaborn_version_str.split('.')
    if int(version_str_parts[1]) < 11:
        kde_axes = sns.distplot(df[col],
                                kde_kws={"label": "Kernel Density", "color": "black"},
                                hist_kws={"label": "Histogram", "color": 'lightsteelblue
    else:
        kde_axes = sns.kdeplot(df[col], color="black", label="Kernel Density")
        sns.histplot(df[col], stat="density", color="lightsteelblue", label="Histogram")

    sns.rugplot(df[col], label="Rug Plot")

    # Add labels and title
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.title('Combined Rug Plot, Histogram, and Kernel Density Plot for '+col)
    plt.legend()
    plt.tight_layout()

    # Show the plot
    plt.show()
```

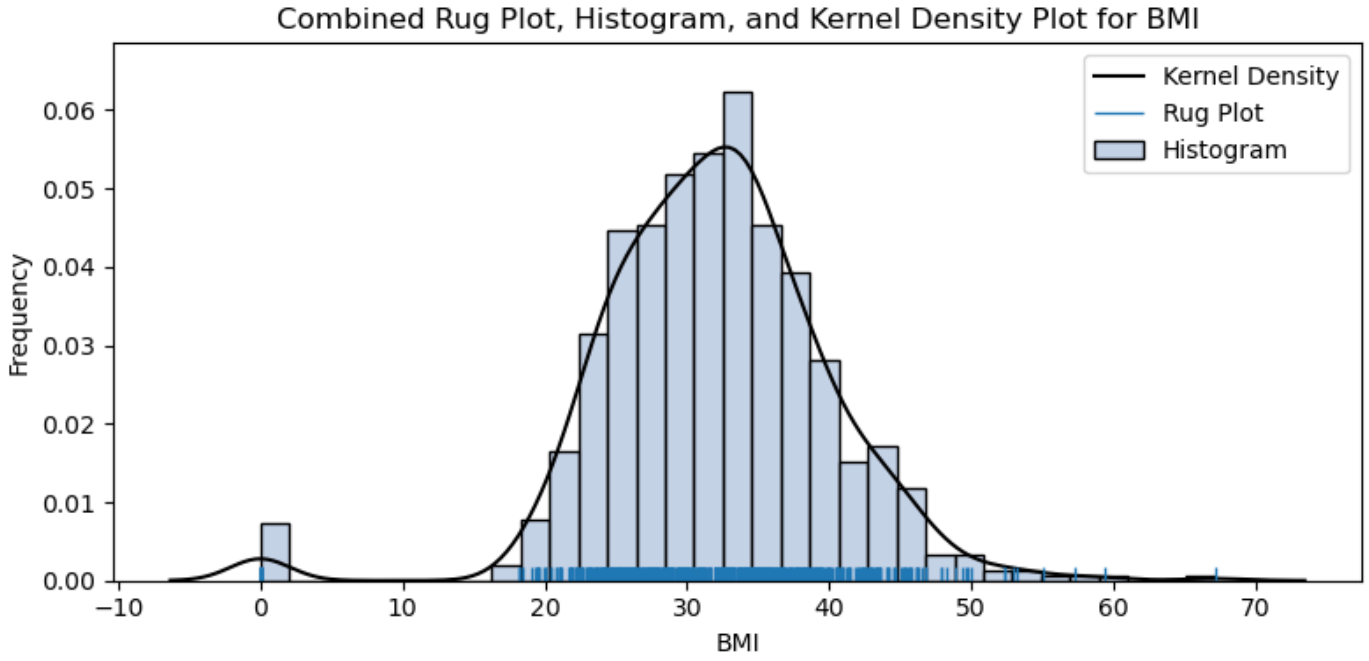

Combined Rug Plot, Histogram, and Kernel Density Plot for Pregnancies
Kernel Density
Rug Plot
Histogram
Frequency
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
Pregnancies

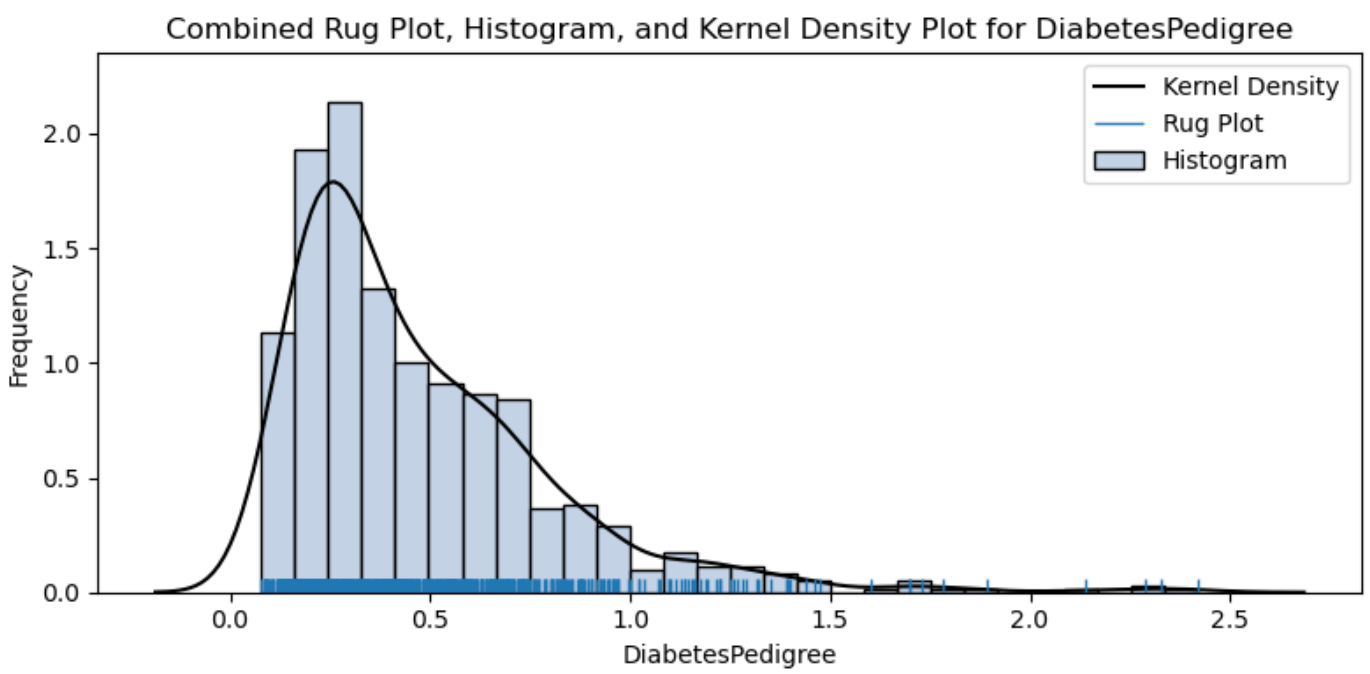

Combined Rug Plot, Histogram, and Kernel Density Plot for Glucose
Kernel Density
Rug Plot
Histogram
Frequency
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
Glucose

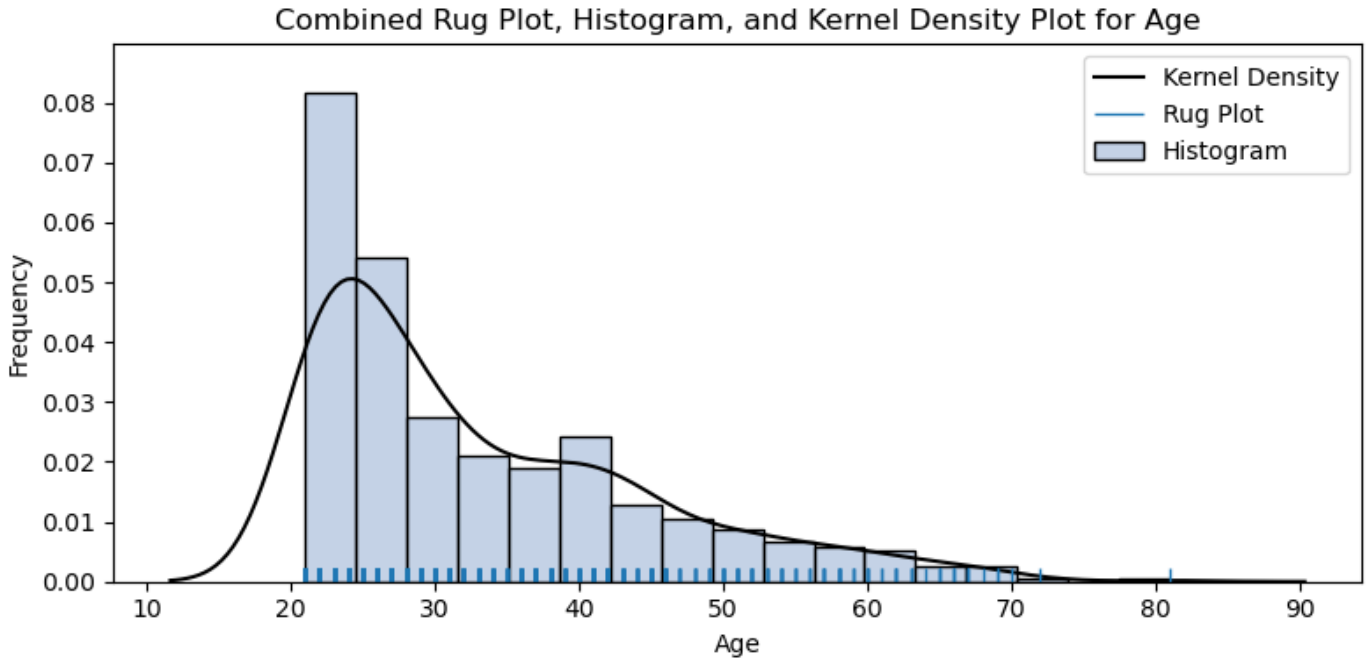

Combined Rug Plot, Histogram, and Kernel Density Plot for BloodPressure
Kernel Density
Rug Plot
Histogram
Frequency
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
−20
0
20
40
60
80
100
120
140
BloodPressure


Combined Rug Plot, Histogram, and Kernel Density Plot for SkinThickness
Kernel Density
Rug Plot
Histogram
Frequency
0.00
0.01
0.02
0.03
0.04
0
20
40
60
80
100
SkinThickness


Combined Rug Plot, Histogram, and Kernel Density Plot for Insulin
Kernel Density
Rug Plot
Histogram
Frequency
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0
200
400
600
800
Insulin

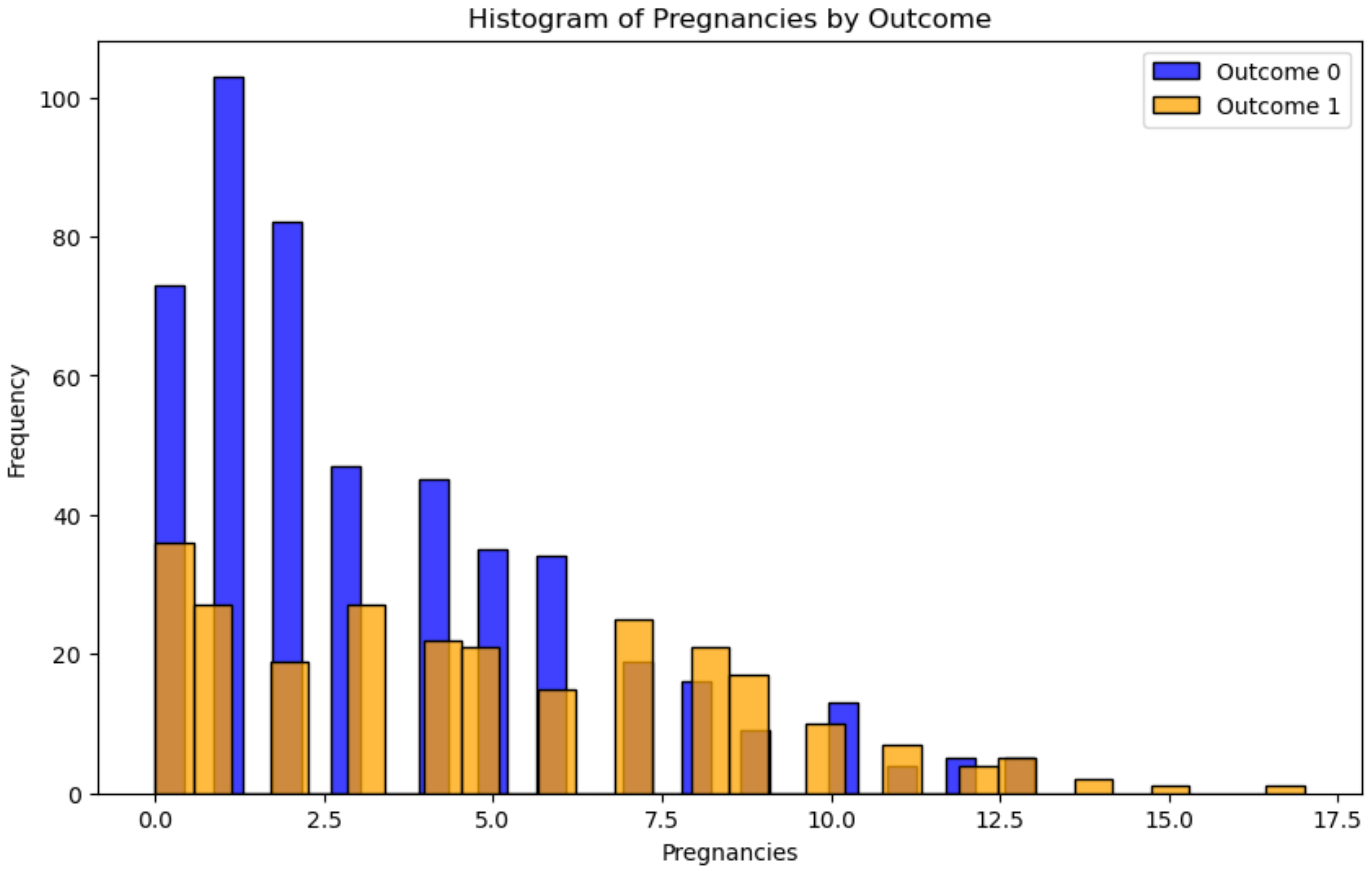

Combined Rug Plot, Histogram, and Kernel Density Plot for BMI
Kernel Density
Rug Plot
Histogram
Frequency
0.00
0.01
0.02
0.03
0.04
0.05
0.06
−10
0
10
20
30
40
50
60
70
BMI

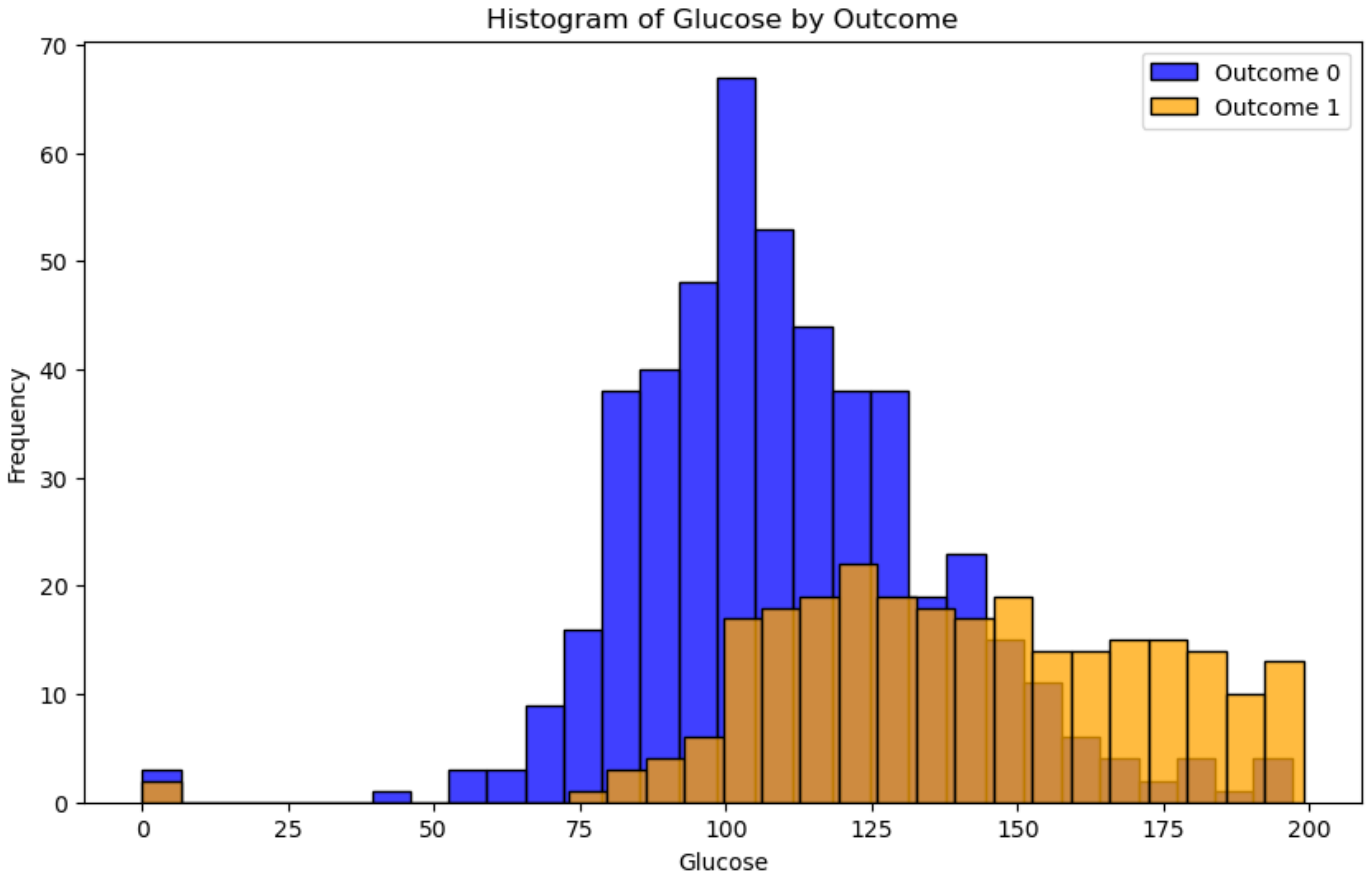


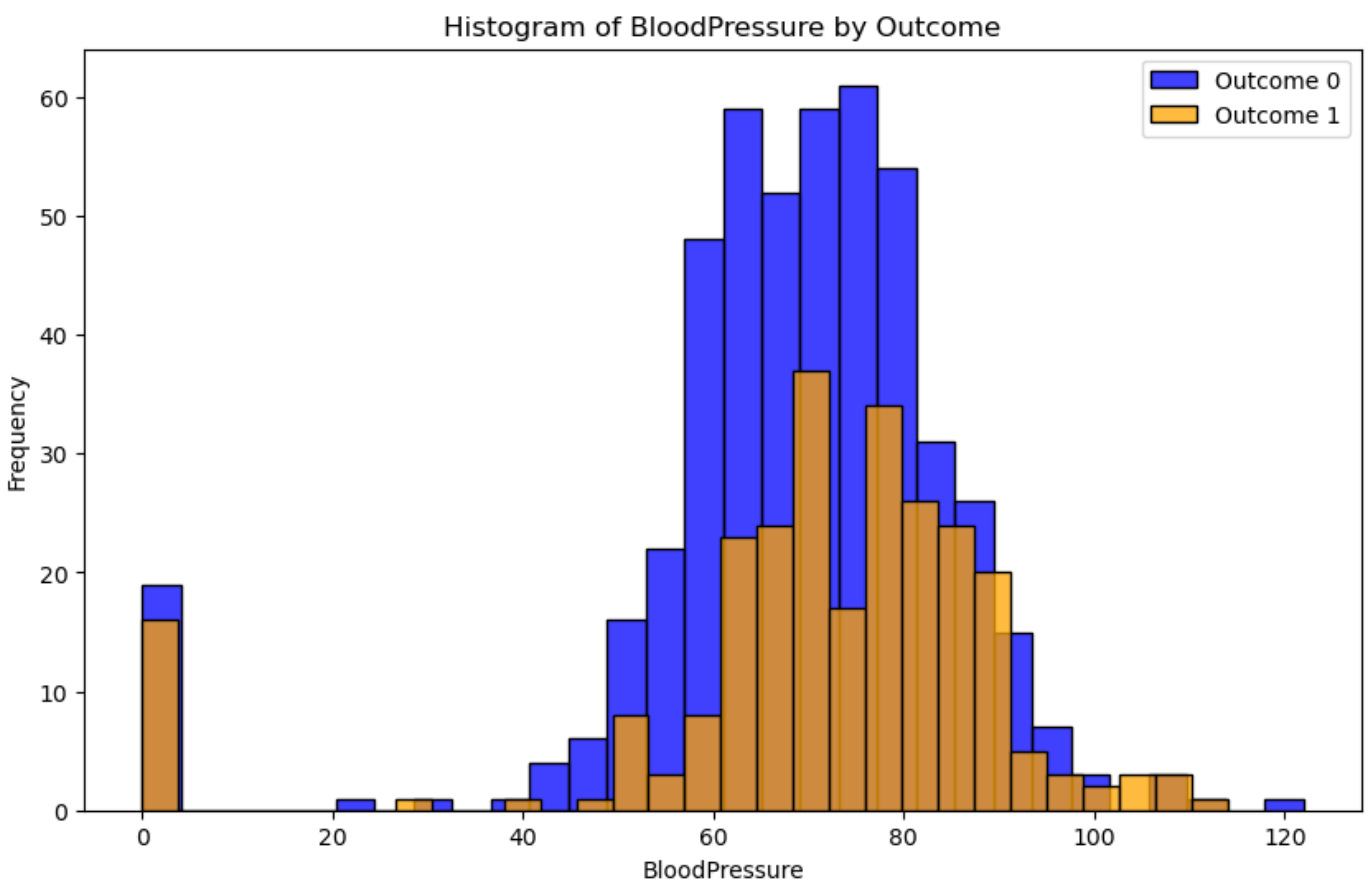


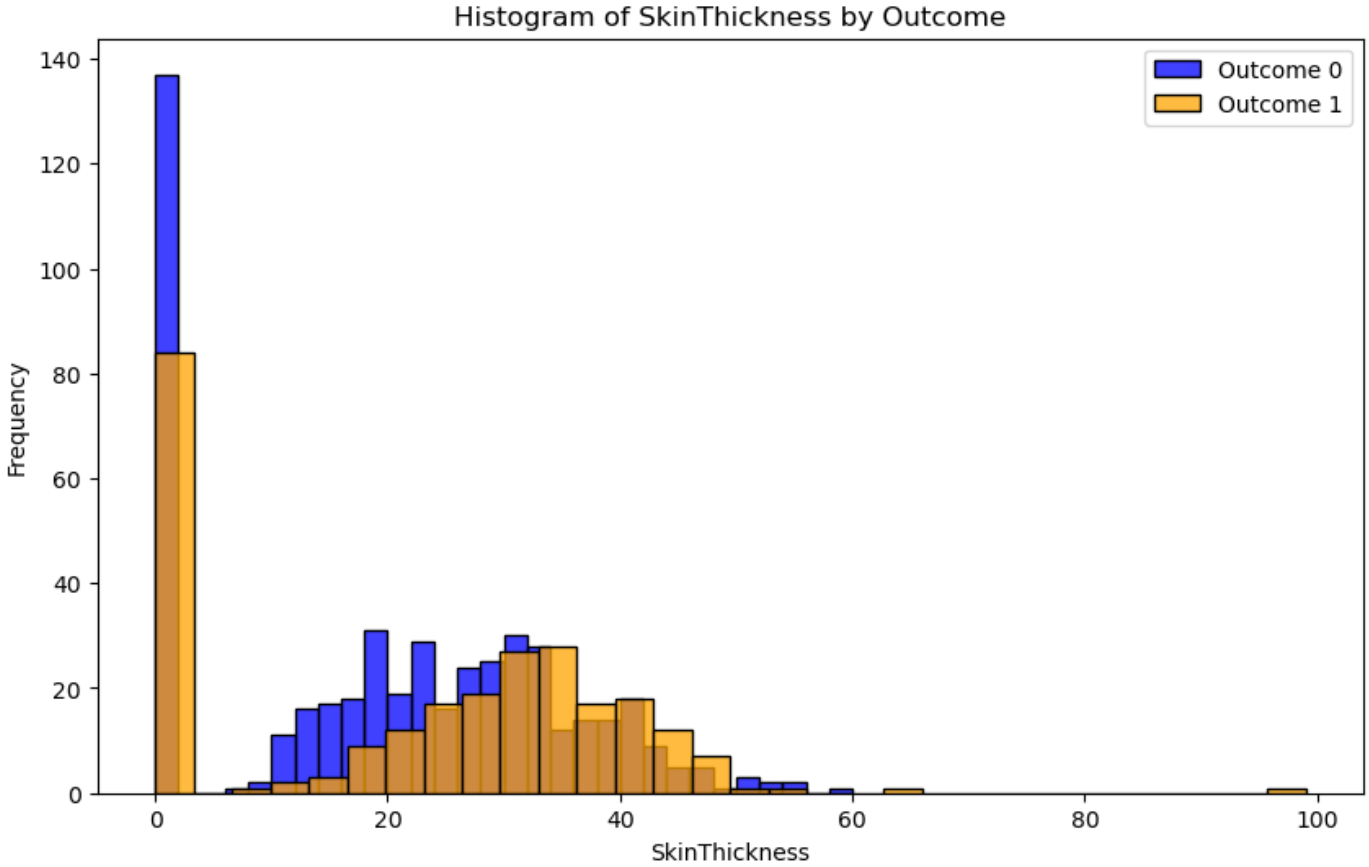
In [9]:

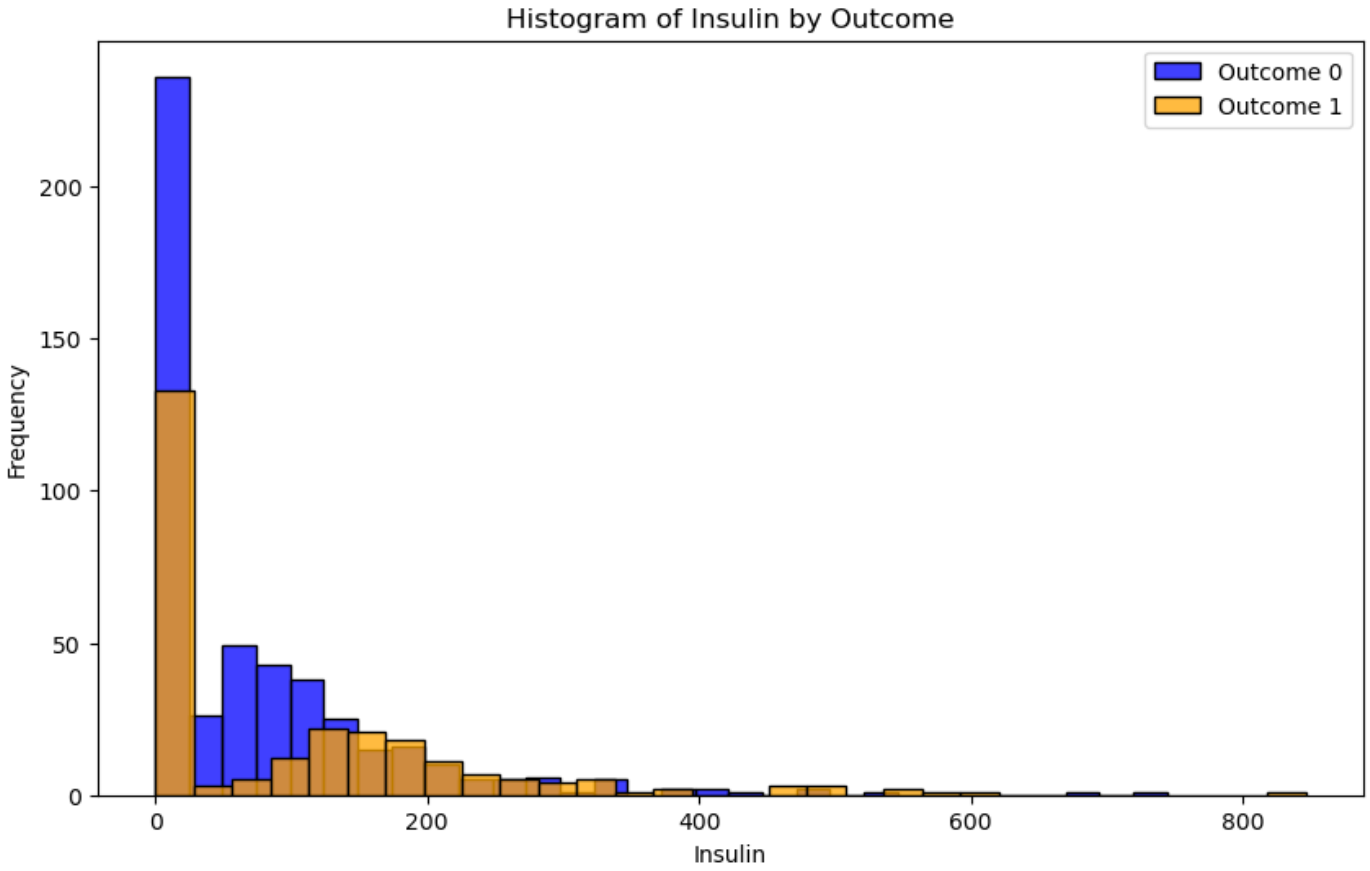
```
outcome_0 = df[df['Outcome'] == 0]
outcome_1 = df[df['Outcome'] == 1]

# Plot histograms
for col in ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','Dia
    plt.figure(figsize=(10, 6))
    sns.histplot(outcome_0[col], bins=30, kde=False, color='blue', label='Outcome 0')
    sns.histplot(outcome_1[col], bins=30, kde=False, color='orange', label='Outcome 1')

    # Add labels and title
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.title('Histogram of ' + col + ' by Outcome')
    plt.legend()
```

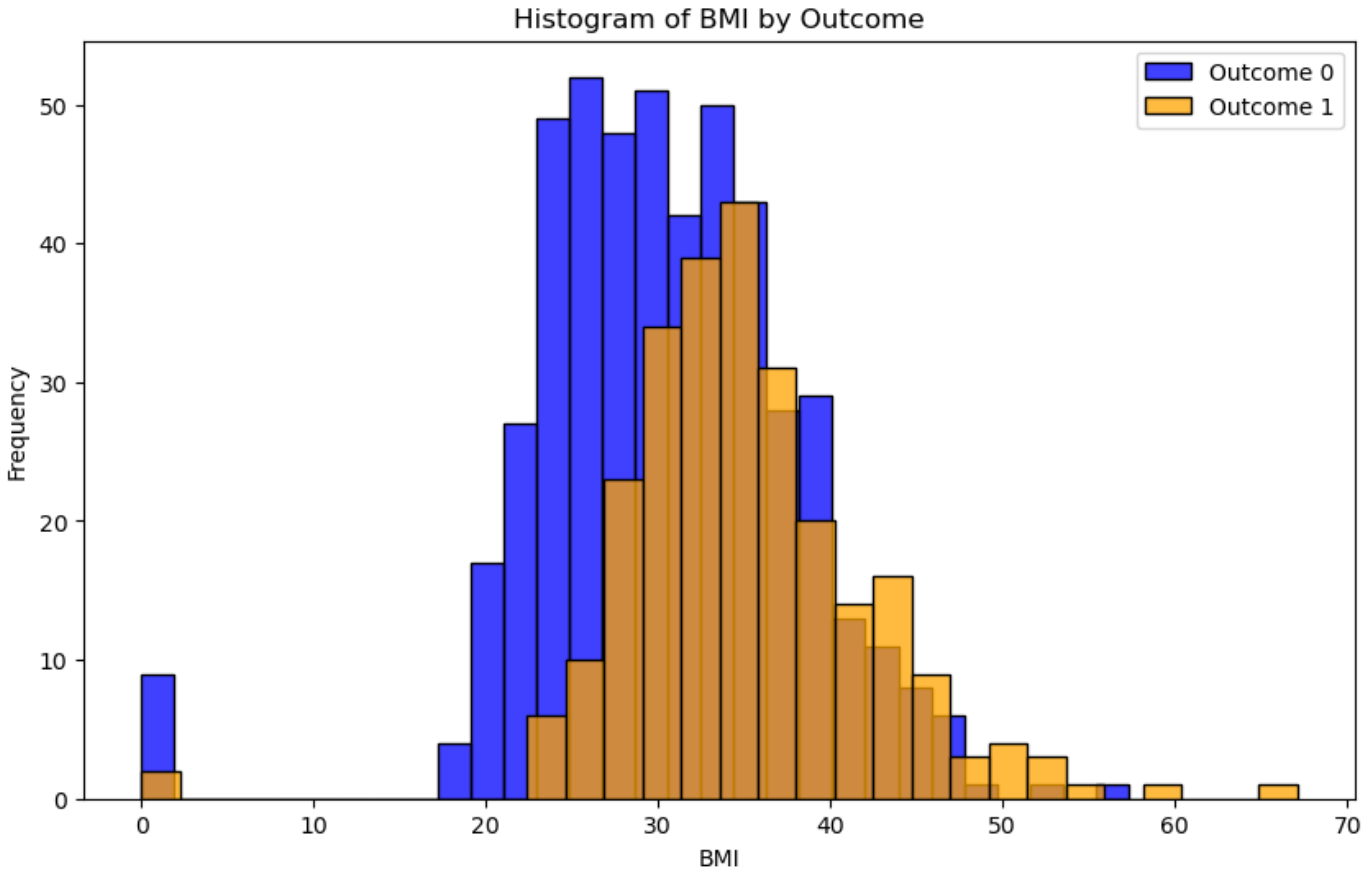

Histogram of Pregnancies by Outcome
Outcome 0
Outcome 1
Frequency
Pregnancies
0
20
40
60
80
100
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

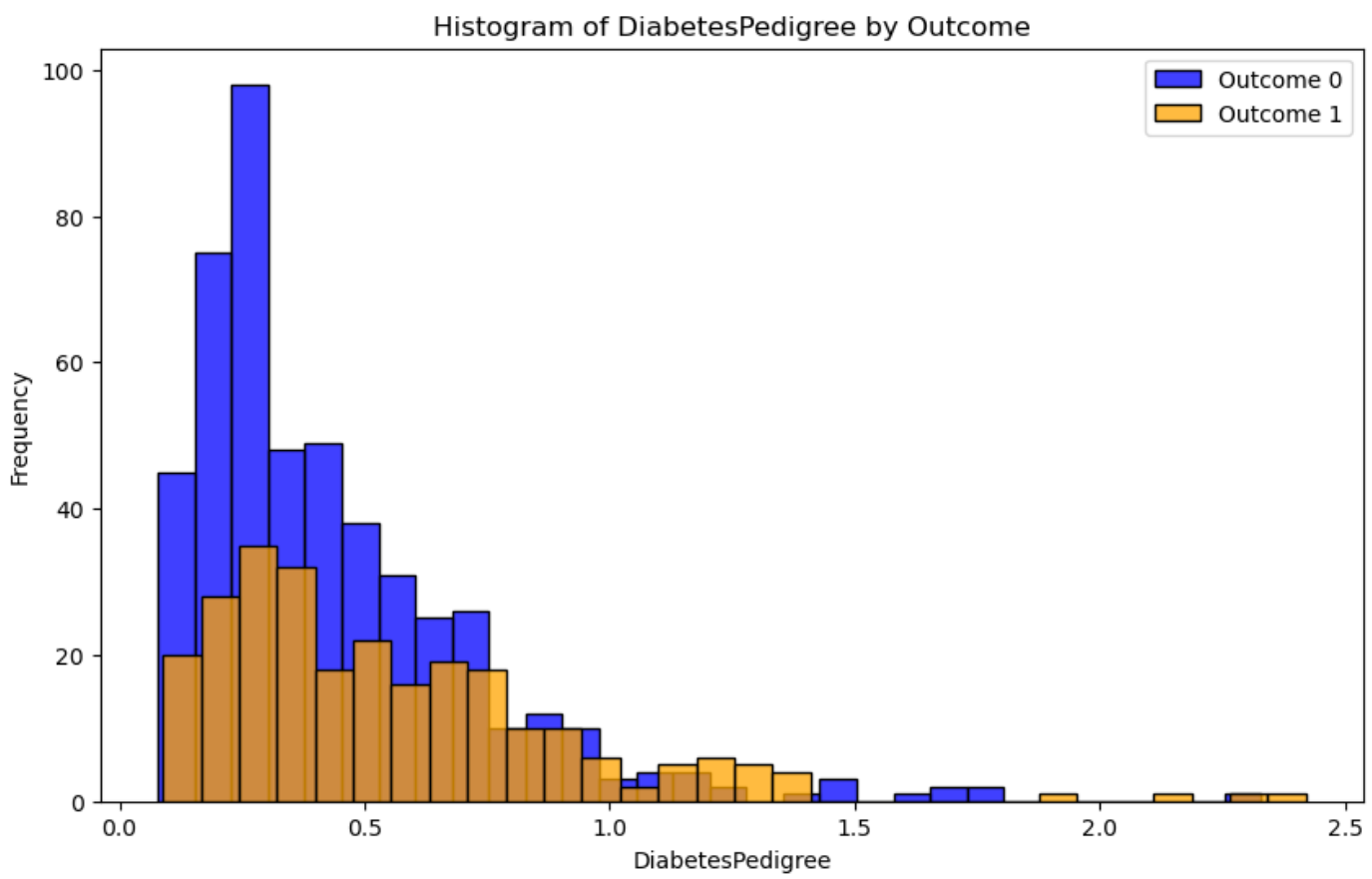

Histogram of Glucose by Outcome
Outcome 0
Outcome 1
Frequency
Glucose
0
10
20
30
40
50
60
70
0
25
50
75
100
125
150
175
200

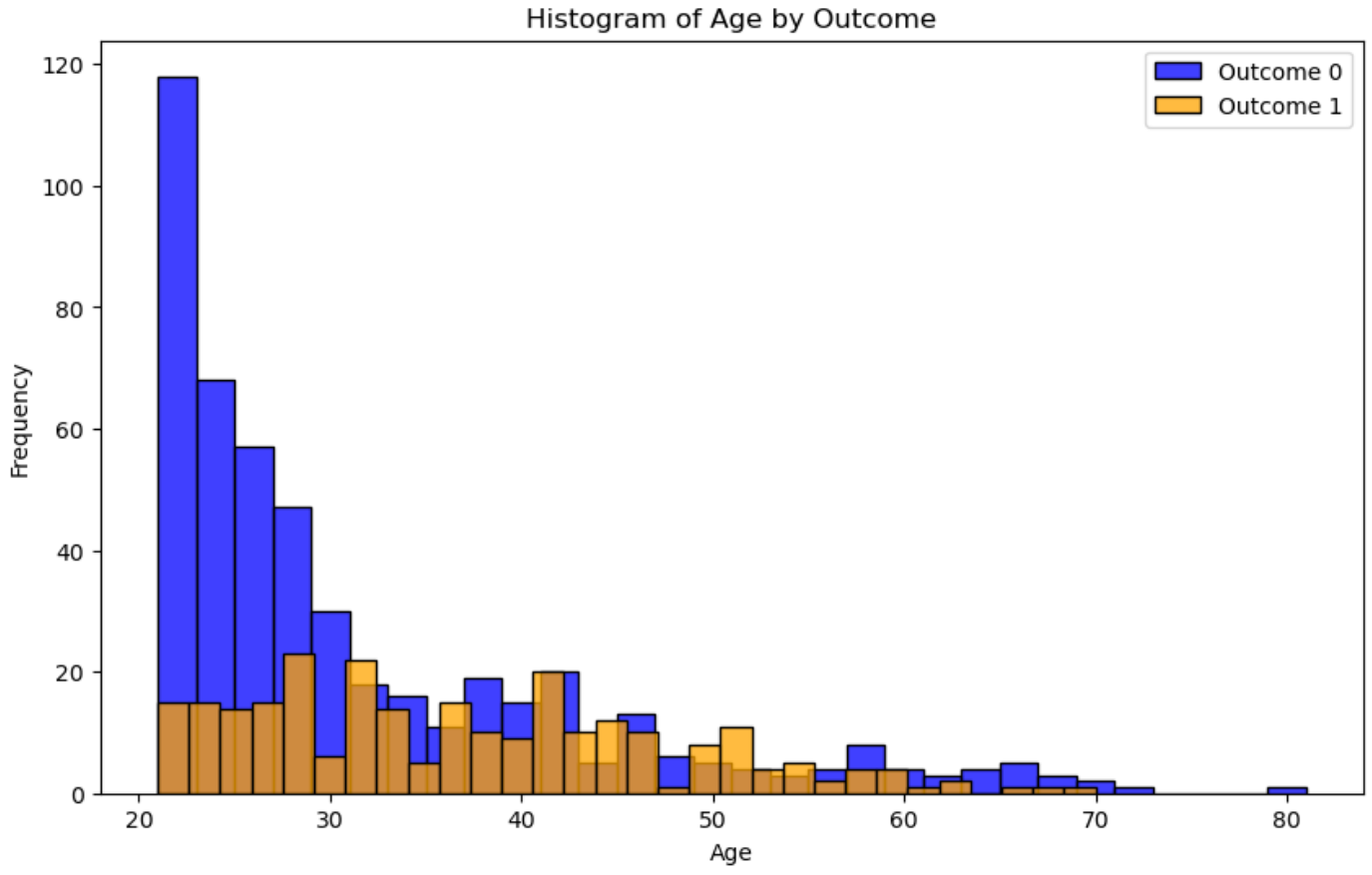

Histogram of BloodPressure by Outcome
Outcome 0
Outcome 1
Frequency
BloodPressure
0
10
20
30
40
50
60
0
20
40
60
80
100
120


Histogram of SkinThickness by Outcome
Outcome 0
Outcome 1
Frequency
SkinThickness
0
20
40
60
80
100
120
140
0
20
40
60
80
100

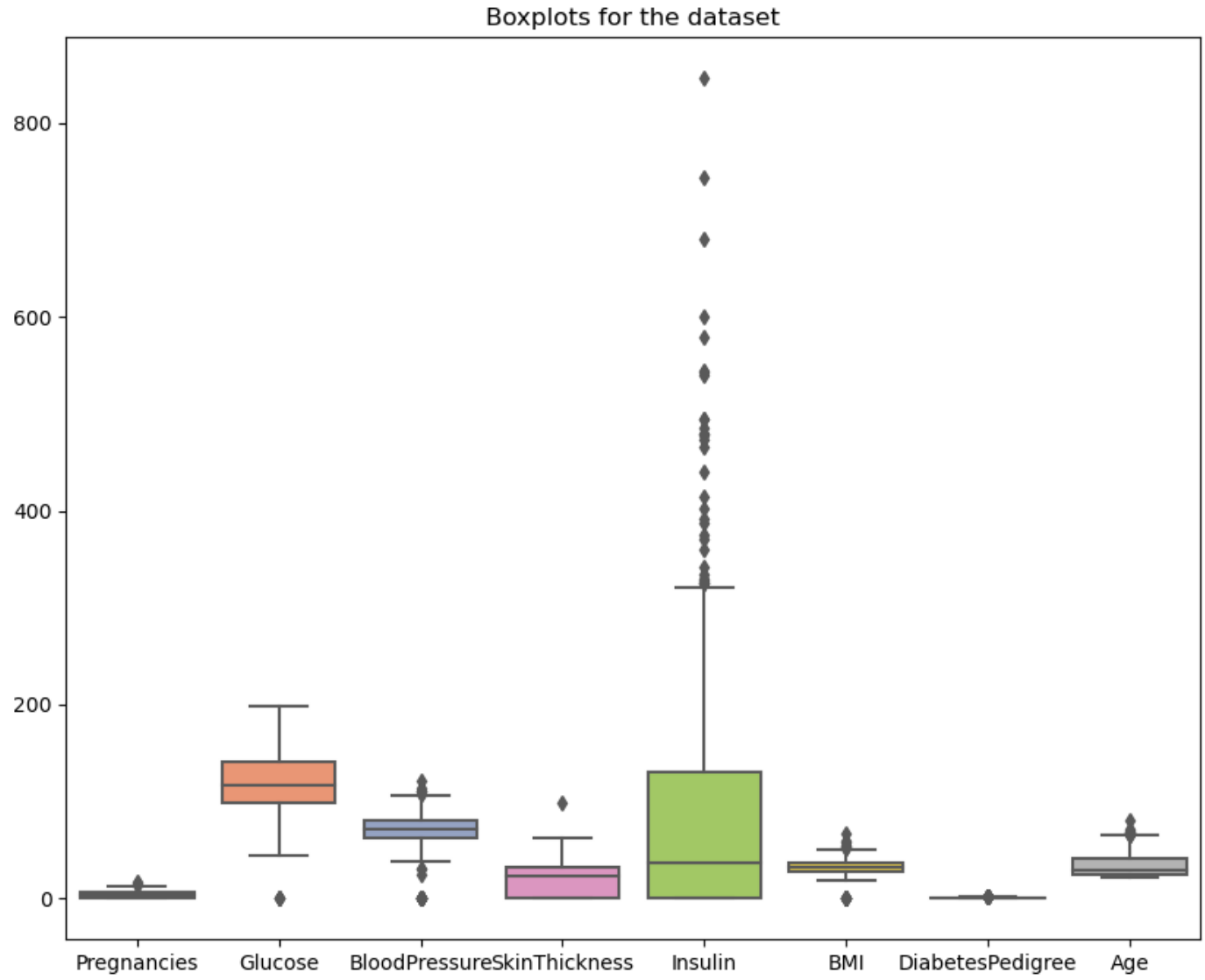

Histogram of Insulin by Outcome
Outcome 0
Outcome 1
Frequency
Insulin
0
50
100
150
200
0
200
400
600
800


Histogram of BMI by Outcome
Outcome 0
Outcome 1
Frequency
BMI
0
10
20
30
40
50
0
10
20
30
40
50
60
70

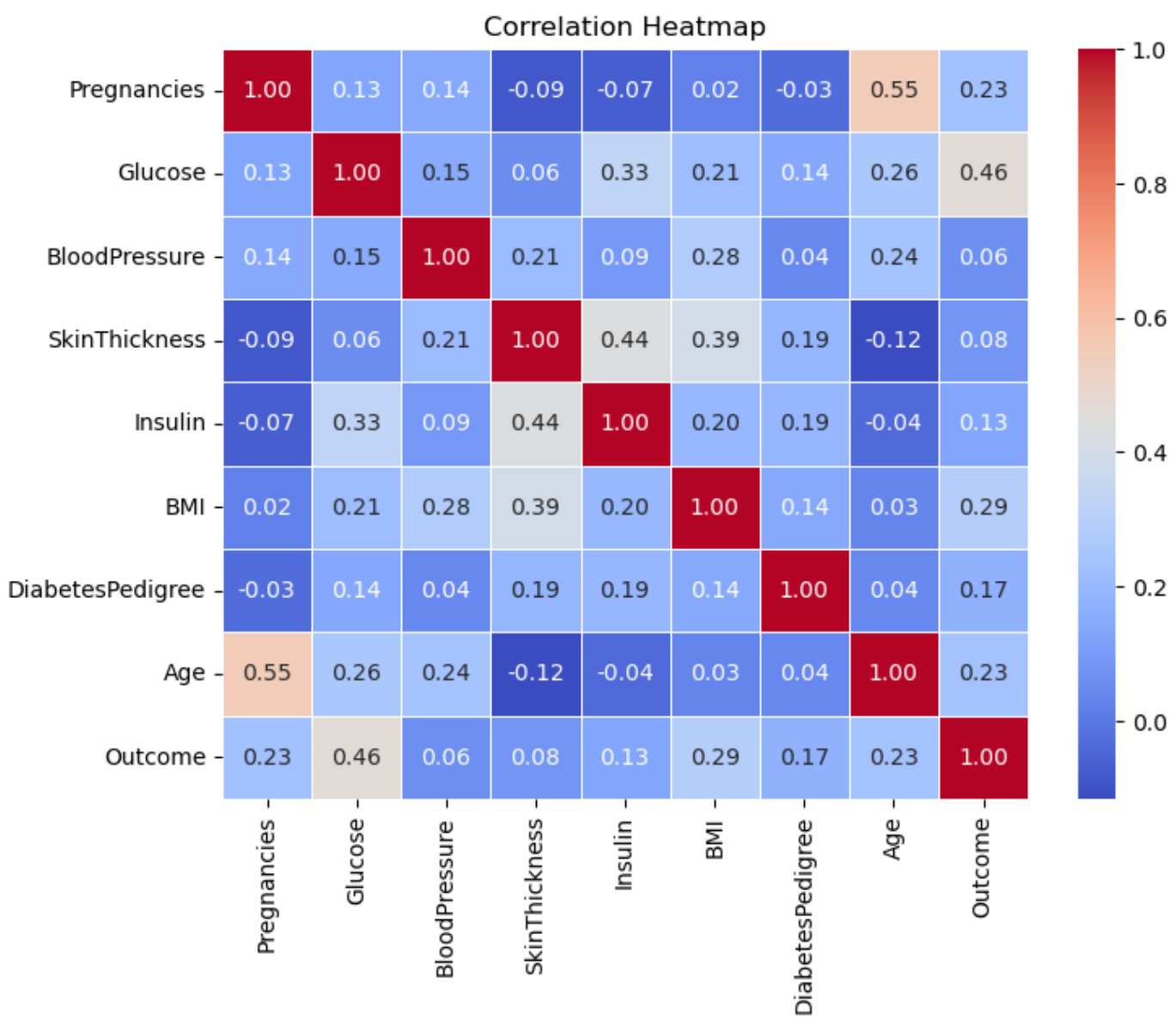




In [10]:
```
#Boxplot for the dataset
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPe
plt.figure(figsize=(10, 8))
sns.boxplot(data=df[col],palette='Set2')
plt.title('Boxplots for the dataset')
plt.show()
```

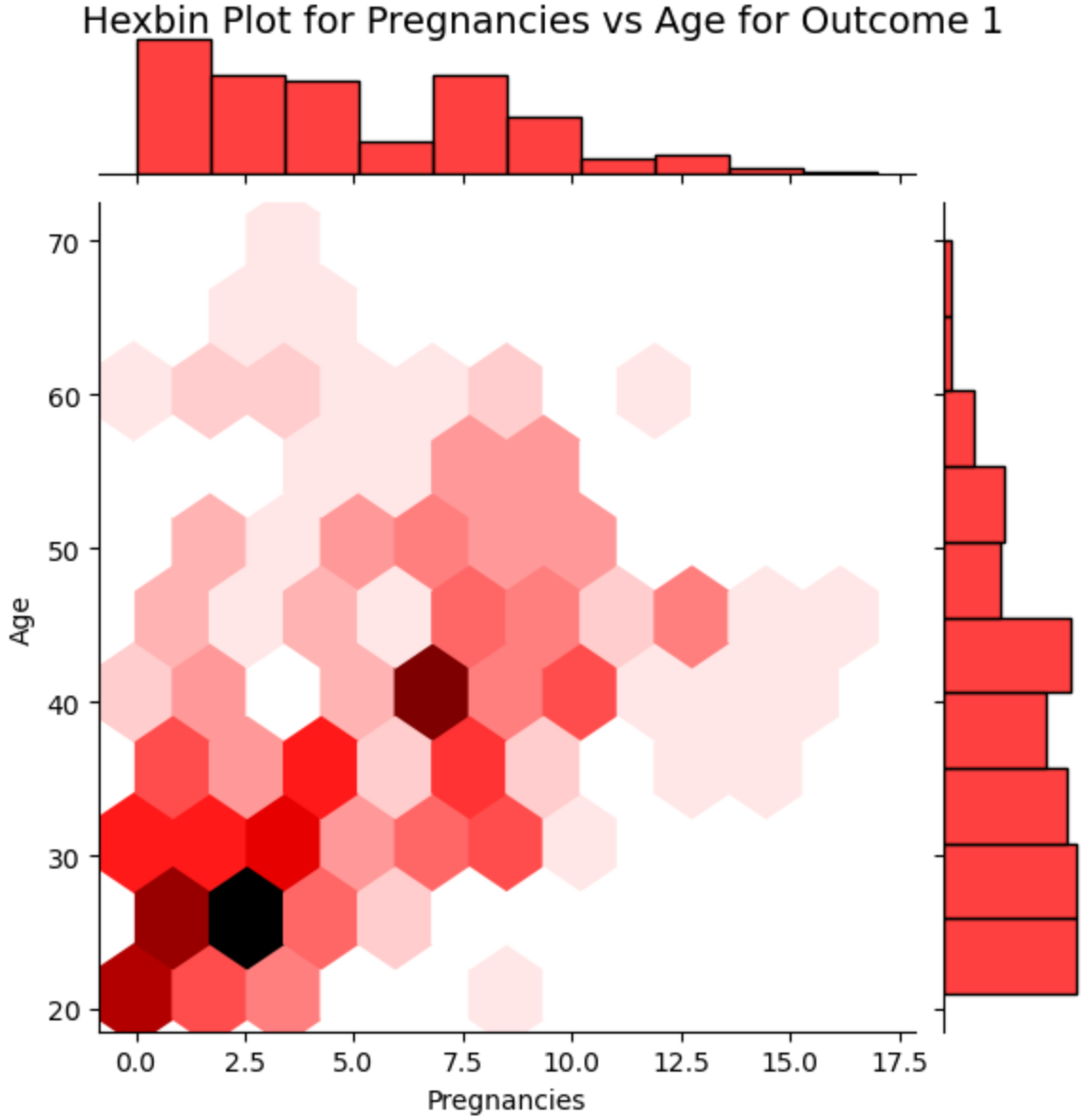


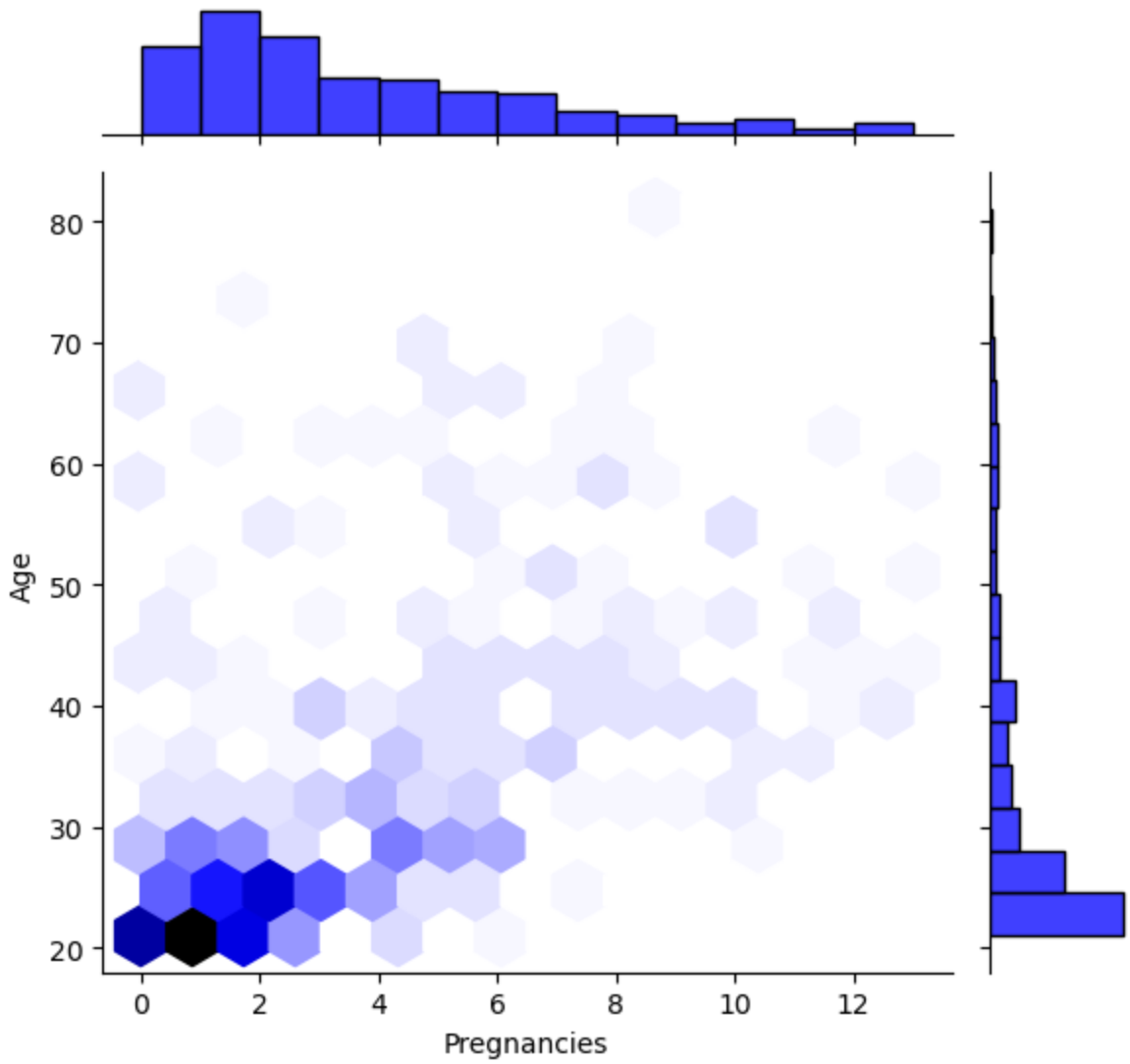
In [11]:
```
#PLotting correlation heatmap for the dataset
corr=df.corr()
plt.figure(figsize=(8, 6))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f', linewidths=.5)
plt.title('Correlation Heatmap')
plt.show()
```



In [12]:
```
#Hexbin Plot for Pregnancy vs Age for each group of Outcome
plt.figure(figsize=(8, 6))
for group, color in zip(df['Outcome'].unique(), ['red', 'blue']):
    sns.jointplot(x='Pregnancies', y='Age', data=df[df['Outcome'] == group], kind='hex',
    plt.xlabel('Pregnancies')
    plt.ylabel('Age')
    plt.suptitle('Hexbin Plot for Pregnancies vs Age for Outcome ' +str(group), x=0.5, y
plt.tight_layout()
plt.show()
```

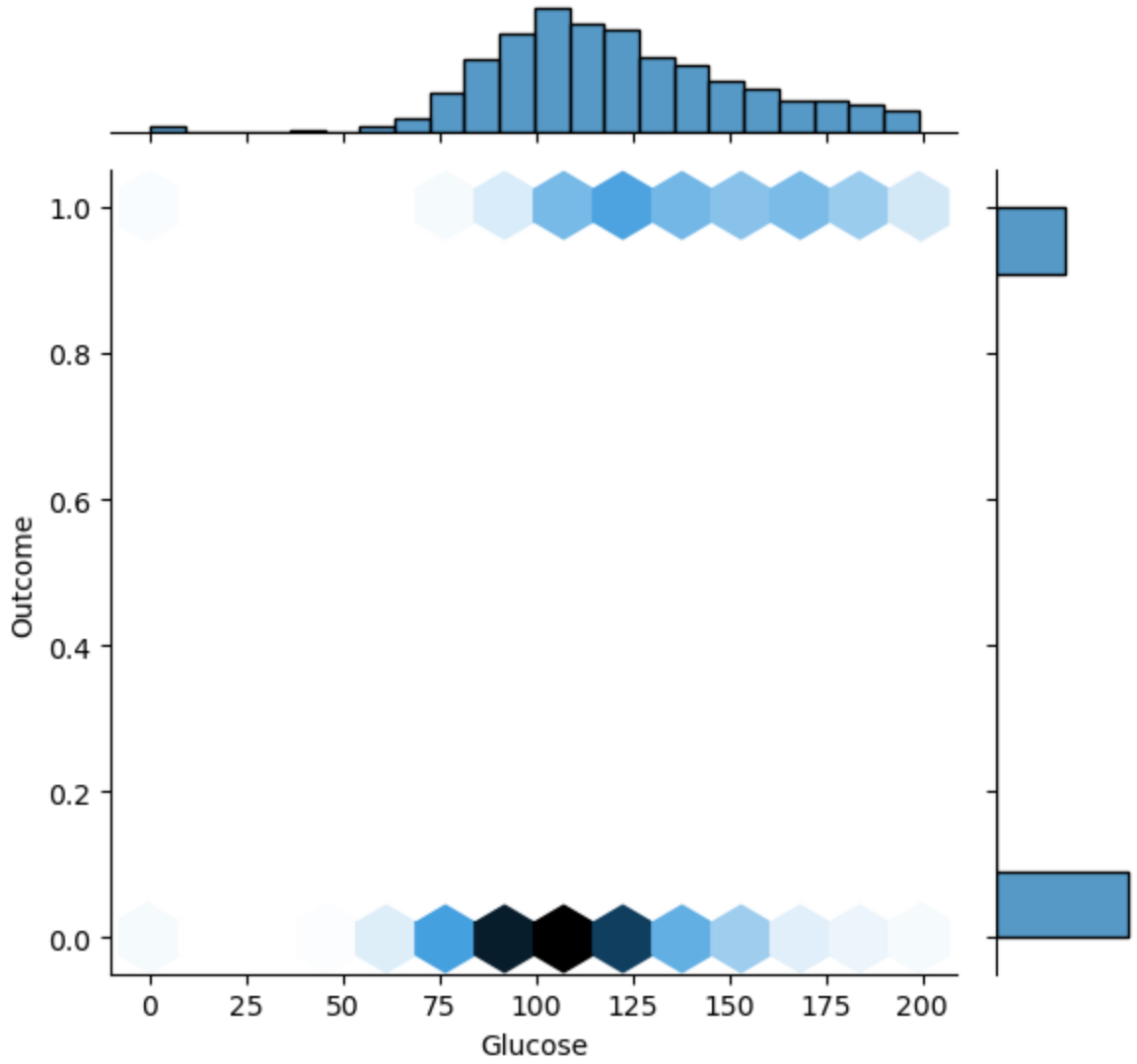
```
<Figure size 800x600 with 0 Axes>
```


Hexbin Plot for Pregnancies vs Age for Outcome 1
Age
Pregnancies
70
60
50
40
30
20
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5



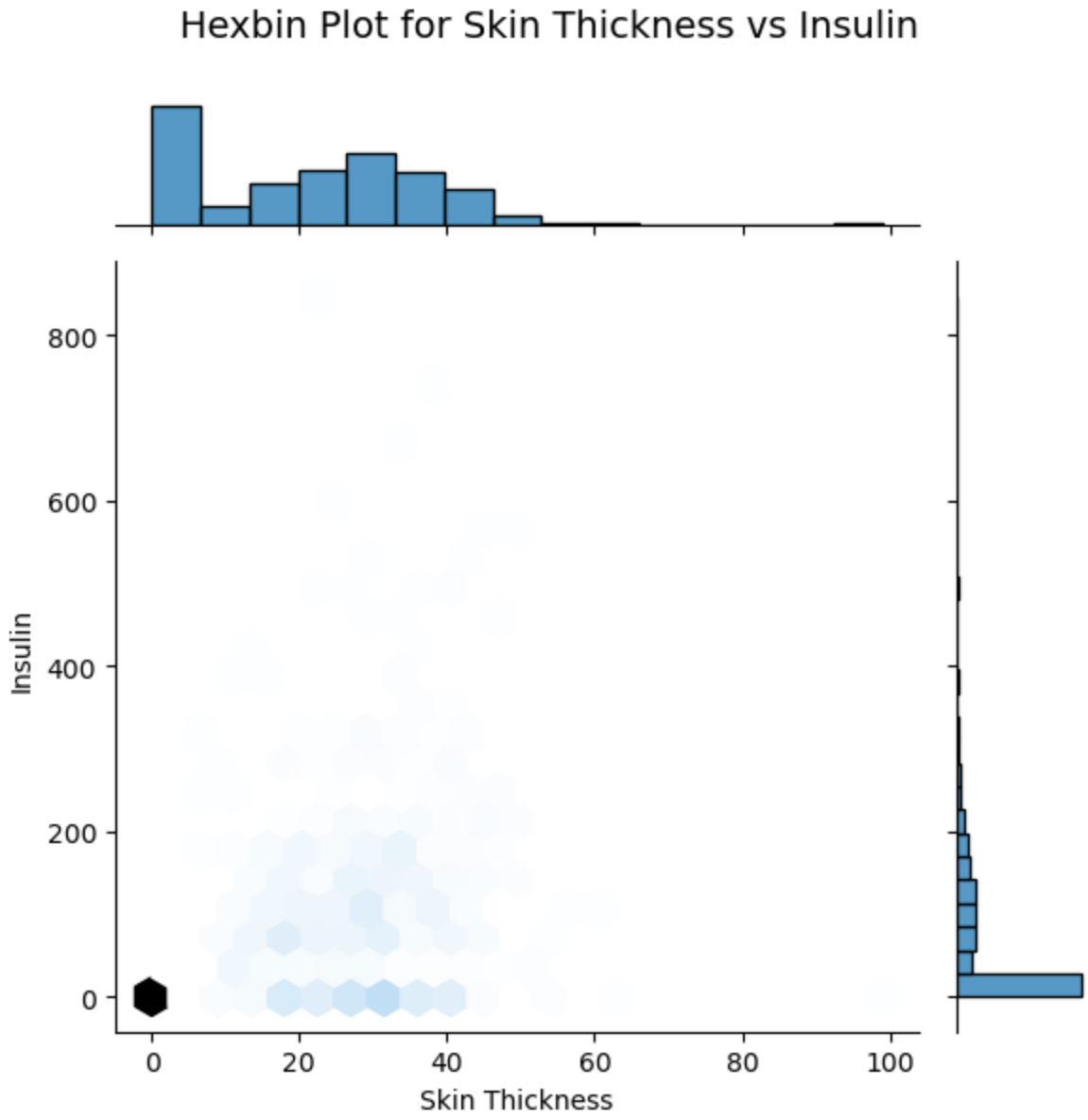
In [13]:
```
#Hexbin Plot for Glucose vs Outcome
plt.figure(figsize=(8, 6))
sns.jointplot(x=df['Glucose'], y=df['Outcome'], kind='hex')
plt.xlabel('Glucose')
plt.ylabel('Outcome')
plt.suptitle('Hexbin Plot for Glucose vs Outcome', x=0.5, y=1, ha='center', fontsize=14)
plt.tight_layout()
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



In [14]:
```
#Hexbin Plot for Skin Thickness vs Insulin
plt.figure(figsize=(8, 6))
sns.jointplot(x=df['SkinThickness'], y=df['Insulin'], kind='hex')
plt.xlabel('Skin Thickness')
plt.ylabel('Insulin')
plt.suptitle('Hexbin Plot for Skin Thickness vs Insulin', x=0.5, y=1, ha='center', fonts
plt.tight_layout()
plt.show()
```

```
<Figure size 800x600 with 0 Axes>
```



Question 3

In [15]:
```
#Create a new column called SevenOrMorePregnancies
df['SevenOrMorePregnancies']=(df['Pregnancies']>=7).astype(int)
df.head(10)
```

Out[15]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outcome	SevenOrM
0	6	148	72	35	0	33.6	0.627	50	1	
1	1	85	66	29	0	26.6	0.351	31	0	
2	8	183	64	0	0	23.3	0.672	32	1	
3	1	89	66	23	94	28.1	0.167	21	0	
4	0	137	40	35	168	43.1	2.288	33	1	
5	5	116	74	0	0	25.6	0.201	30	0	
6	3	78	50	32	88	31.0	0.248	26	1	
7	10	115	0	0	0	35.3	0.134	29	0	
8	2	197	70	45	543	30.5	0.158	53	1	
9	8	125	96	0	0	0.0	0.232	54	1	

In [16]:
```
#Splitting the dataset into training and testing set
X = df[['SevenOrMorePregnancies']]
Y = df['Outcome']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42
```

In [17]:
```
#Using Logistic Regression and fitting the model
model = LogisticRegression()
model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)

accuracy = accuracy_score(Y_test, Y_pred)

print('Accuracy: '+ str(accuracy_score(Y_test, Y_pred)))
print('F-score: ' + str(f1_score(Y_test, Y_pred)))
```

```
Accuracy: 0.6866666666666666
F-score: 0.4050632911392405
```

In [18]:
```
#Finding the probability of getting diabetes

#Storing the probability for the second class which is when Outcome is 1 into p
p=model.predict_proba(X_test)[:, 1]

#Probability of getting diabetes with six or fewer pregnancies
six_less=p[X_test['SevenOrMorePregnancies'] == 0].mean()

#Probability of getting diabetes with seven or more pregnancies
seven_more = p[X_test['SevenOrMorePregnancies'] == 1].mean()


print('Probability of diabetes with six or fewer pregnancies: ' + str(six_less))
print('Probability of diabetes with seven or more pregnancies: ' + str(seven_more))
```

```
Probability of diabetes with six or fewer pregnancies: 0.29464902942023524
Probability of diabetes with seven or more pregnancies: 0.5787003497647526
```

Question 4

Preprocessing of PimaDiabetes.csv

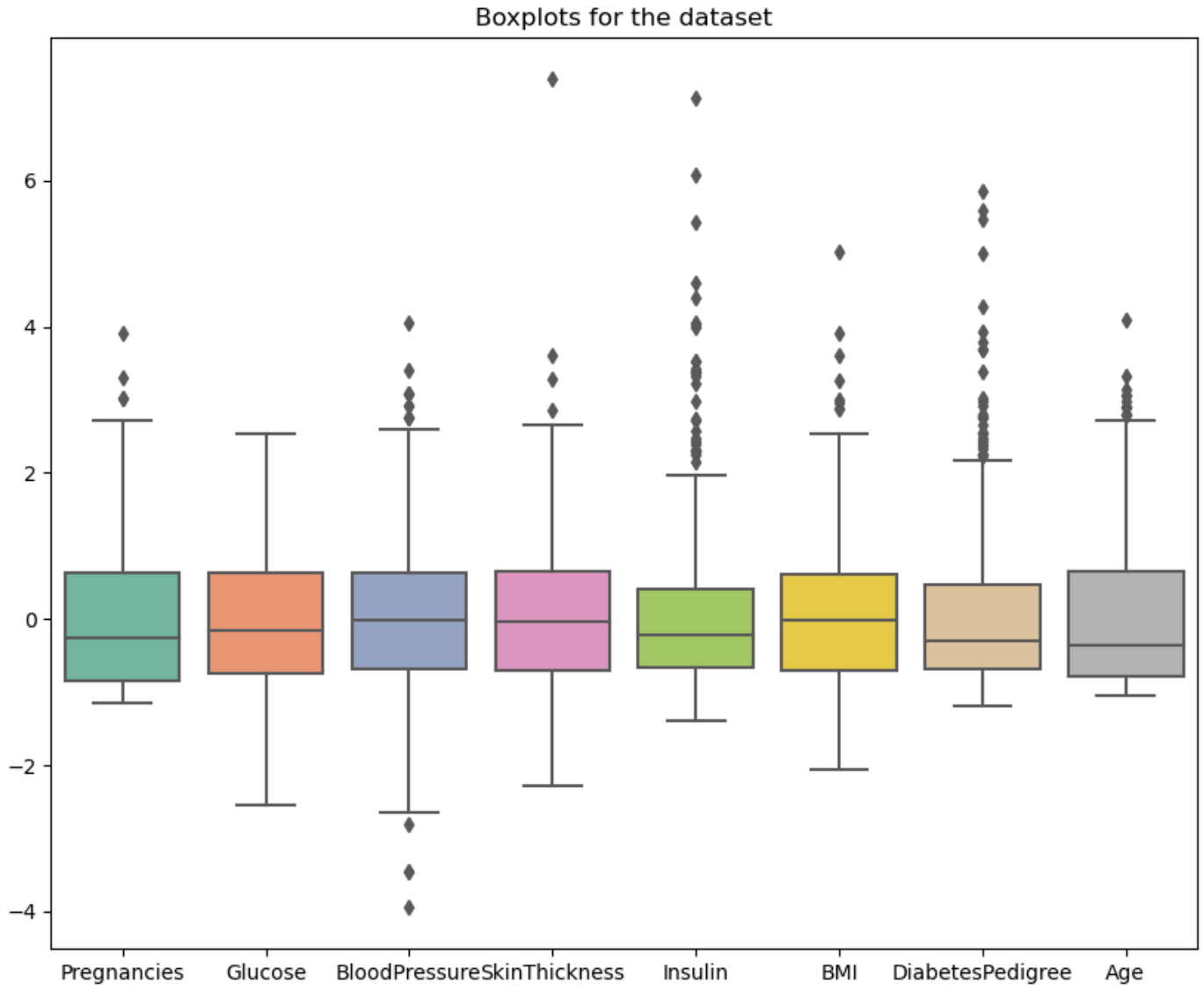
In [19]:
```
#Columns to fill 0 values
col = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI' ]

# 0 is replaced with NaN
df[col] = df[col].replace(0, np.nan)

#Copy of df with the selected columns
i = df[col].copy()

# KNN Imputer
knn = KNNImputer(n_neighbors=5)
i_arr = knn.fit_transform(i)

# Convert the array to dataframe
imputed = pd.DataFrame(i_arr, columns=col)

# Update df with the new values from dataframe imputed
df.loc[:, col] = imputed

df.describe()
```

Out[19]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Ag
count	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.000000	750.0000
mean	3.844000	121.612000	72.304533	28.782933	149.704267	32.434507	0.473544	33.1666
std	3.370085	30.480089	12.237632	9.508398	97.765886	6.912747	0.332119	11.7088
min	0.000000	44.000000	24.000000	7.000000	14.000000	18.200000	0.078000	21.0000
25%	1.000000	99.000000	64.000000	22.000000	85.300000	27.500000	0.244000	24.0000
50%	3.000000	117.000000	72.000000	28.500000	129.200000	32.300000	0.377000	29.0000
75%	6.000000	141.000000	80.000000	35.000000	189.500000	36.600000	0.628500	40.7500
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.0000

In [21]:
```
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPe
X=df[col]

#StandardScaler
s=StandardScaler()
X_scale = s.fit_transform(X)

# Replace the values in df
df[col] = X_scale
```

In [22]:
```
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPe
plt.figure(figsize=(10, 8))
sns.boxplot(data=df[col],palette='Set2')
plt.title('Boxplots for the dataset')
plt.show()
```



In [23]:
```
def outliers(col):
    Q1 = col.quantile(0.25)
    Q3 = col.quantile(0.75)
    IQR = Q3 - Q1
    l=Q1 - 1.5 * IQR
    u=Q3 + 1.5 * IQR
    return col.apply(lambda x: x if (x>=l and x<=u) else None)
```

In [24]:
```
col=['Pregnancies','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigree','A

for c in col:
    df[c] = outliers(df[c])
df = df.dropna()
```

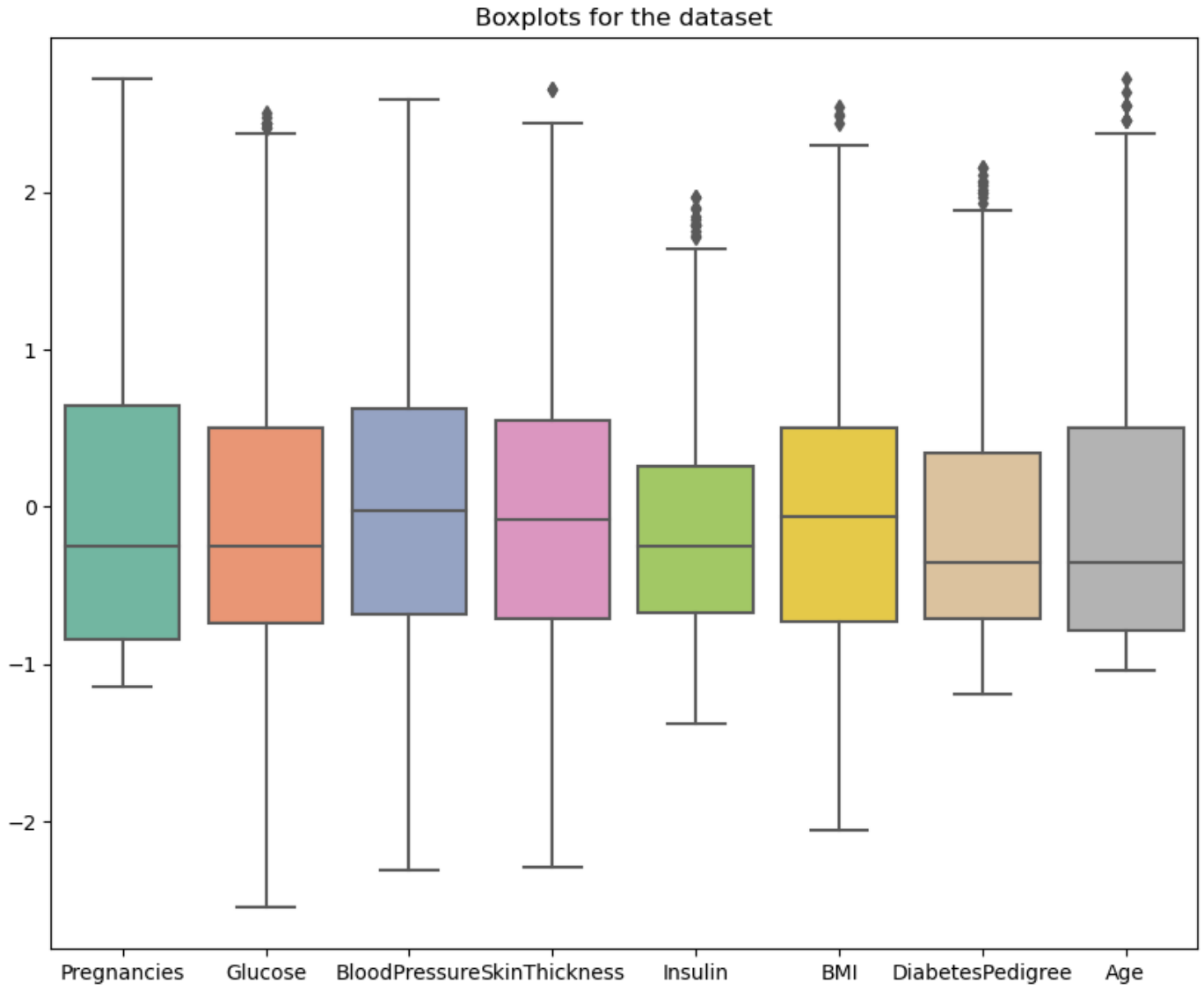
In [25]:
```
df.describe()
```

Out[25]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Ag
count	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	660.000000	660.0000
mean	-0.021163	-0.086996	-0.029981	-0.056533	-0.144136	-0.060981	-0.137816	-0.0793
std	0.966896	0.946568	0.896331	0.938913	0.703973	0.927727	0.750830	0.9074
min	-1.141385	-2.548017	-2.314453	-2.292444	-1.378744	-2.060542	-1.191764	-1.0397
25%	-0.844459	-0.742356	-0.679059	-0.713838	-0.673017	-0.728779	-0.709688	-0.7834
50%	-0.250606	-0.249903	-0.024902	-0.082396	-0.252857	-0.062898	-0.354157	-0.3560

75%	0.640173	0.505191	0.629256	0.549046	0.258910	0.501654	0.338827	0.4985
max	2.718658	2.507833	2.591728	2.653853	1.968213	2.542725	2.161676	2.7205

In [26]:
```
#Boxplot after Standardizing the dataset and removing the Outliers
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPe
plt.figure(figsize=(10, 8))
sns.boxplot(data=df[col],palette='Set2')
plt.title('Boxplots for the dataset')
plt.show()
```



Preprocessing of ToPredict.csv

In [27]:
```
test=pd.read_csv('ToPredict.csv')
test.describe()
```

Out[27]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
count	5.000000	5.00000	5.000000	5.000000	5.000000	5.000000	5.000000	5.000000
mean	3.200000	140.00000	75.200000	27.800000	116.800000	34.380000	0.466200	29.200000
std	3.114482	28.62691	9.757049	17.268468	222.128791	6.803822	0.410425	9.093954
min	0.000000	108.00000	62.000000	0.000000	0.000000	26.000000	0.222000	22.000000
25%	1.000000	121.00000	70.000000	24.000000	0.000000	31.200000	0.223000	25.000000

50%	3.000000	136.00000	78.000000	32.000000	0.000000	32.400000	0.261000	26.000000
75%	4.000000	154.00000	78.000000	39.000000	74.000000	39.000000	0.443000	28.000000
max	8.000000	181.00000	88.000000	44.000000	510.000000	43.300000	1.182000	45.000000

In [28]:
```
test.head(10)
```

Out[28]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
0	4	136	70	0	0	31.2	1.182	22
1	1	121	78	39	74	39.0	0.261	28
2	3	108	62	24	0	26.0	0.223	25
3	0	181	88	44	510	43.3	0.222	26
4	8	154	78	32	0	32.4	0.443	45

In [29]:
```
col=['SkinThickness', 'Insulin' ]

# 0 is replaced with NaN
test[col] = test[col].replace(0, np.nan)

#Copy of test with the selected columns
i = test[col].copy()

# KNN Imputer
knn = KNNImputer(n_neighbors=5)
i_arr = knn.fit_transform(i)

# Convert the array to dataframe
imputed = pd.DataFrame(i_arr, columns=col)

# Update test with the new values from dataframe imputed
test.loc[:, col] = imputed

test.describe()
```

Out[29]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
count	5.000000	5.00000	5.000000	5.00000	5.000000	5.000000	5.000000	5.000000
mean	3.200000	140.00000	75.200000	34.75000	292.000000	34.380000	0.466200	29.200000
std	3.114482	28.62691	9.757049	7.52911	154.149278	6.803822	0.410425	9.093954
min	0.000000	108.00000	62.000000	24.00000	74.000000	26.000000	0.222000	22.000000
25%	1.000000	121.00000	70.000000	32.00000	292.000000	31.200000	0.223000	25.000000
50%	3.000000	136.00000	78.000000	34.75000	292.000000	32.400000	0.261000	26.000000
75%	4.000000	154.00000	78.000000	39.00000	292.000000	39.000000	0.443000	28.000000
max	8.000000	181.00000	88.000000	44.00000	510.000000	43.300000	1.182000	45.000000

In [30]:
```
test.head()
```

Out[30]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
0	4	136	70	34.75	292.0	31.2	1.182	22
1	1	121	78	39.00	74.0	39.0	0.261	28

2	3	108	62	24.00	292.0	26.0	0.223	25
3	0	181	88	44.00	510.0	43.3	0.222	26
4	8	154	78	32.00	292.0	32.4	0.443	45

In [31]:
```
col=['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPe
X=test[col]

#StandardScaler
s=StandardScaler()
X_scale = s.fit_transform(X)

# Replace the values in df
test[col] = X_scale
```

In [32]:
```
test.head()
```

Out[32]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age
0	0.287183	-0.156221	-0.595854	0.000000	0.000000	-0.522552	1.949902	-0.885186
1	-0.789754	-0.742052	0.320844	0.631103	-1.581139	0.759179	-0.558983	-0.147531
2	-0.071796	-1.249771	-1.512553	-1.596319	0.000000	-1.377038	-0.662498	-0.516359
3	-1.148733	1.601269	1.466718	1.373577	1.581139	1.465774	-0.665222	-0.393416
4	1.723100	0.546775	0.320844	-0.408361	0.000000	-0.325362	-0.063199	1.942492

Training the model

In [33]:
```
#Splitting data into training and testing set
col=['Glucose','Insulin','Age','BMI','DiabetesPedigree','SkinThickness','BloodPressure',
X=df[col]
Y=df['Outcome']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42

#Logistic Regression model
model=LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

print('Accuracy: ' + str(accuracy_score(Y_test, Y_pred)))
print('F-score: ' + str(f1_score(Y_test, Y_pred)))
```

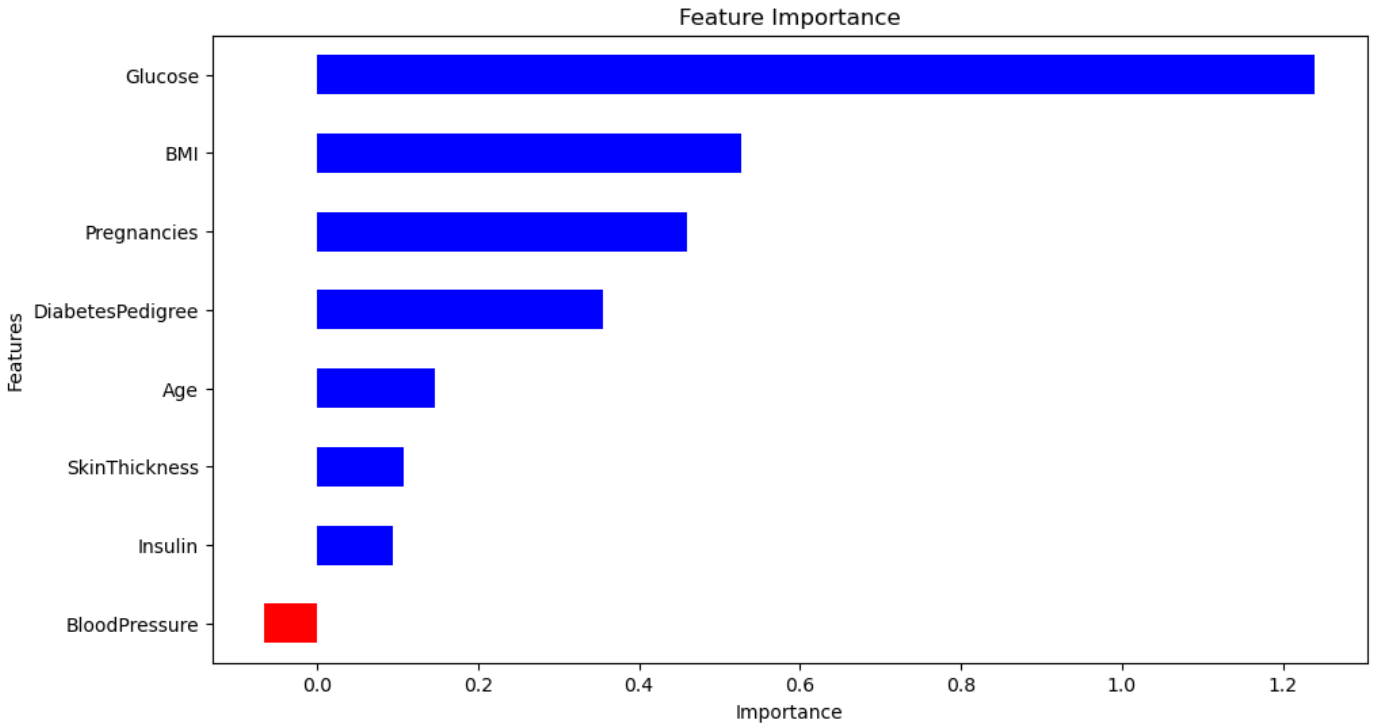
```
Accuracy: 0.7121212121212122
F-score: 0.5128205128205129
```

In [34]:
```
#Getting the coefficients
c=list(model.coef_[0])
labels=list(X_train.columns)

#Creating a dataframe that stores the coefficients
f=pd.DataFrame()
f['Features'] = labels
f['importance'] = c

f.sort_values(by=['importance'], ascending=True, inplace=True)
f['positive'] = f['importance'] > 0
f.set_index('Features', inplace=True)

#Plotting the graph for Feature Importance
```

```
f.importance.plot(kind='barh', figsize=(11, 6),color = f.positive.map({True: 'blue', Fal
plt.title('Feature Importance')
plt.xlabel('Importance')
```

Out[34]:

```
Text(0.5, 0, 'Importance')
```



In [35]:

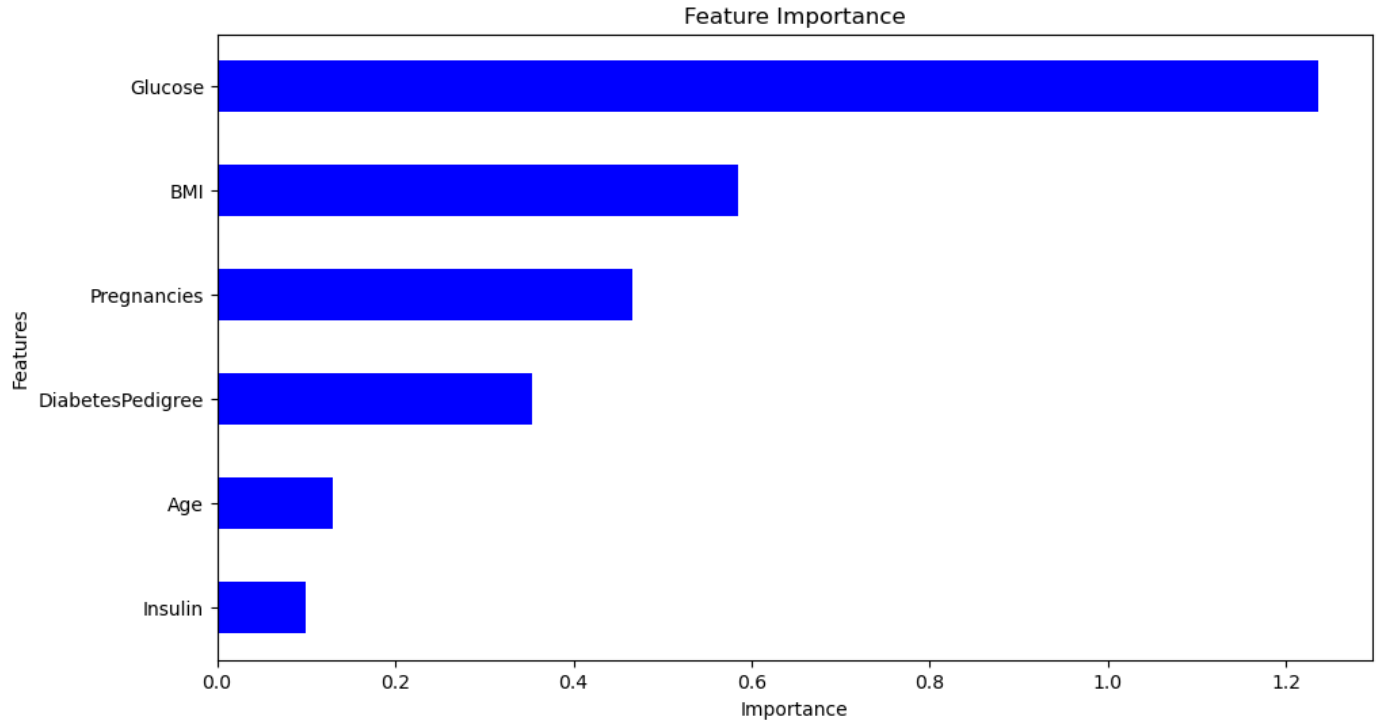
```
#New set of variables after removing Blood Pressure
col=['Glucose','Pregnancies','BMI','DiabetesPedigree','Age','Insulin']
X=df[col]
Y=df['Outcome']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42

#Logistic Regression Model
model=LogisticRegression()
model.fit(X_train,Y_train)
Y_pred = model.predict(X_test)

print('Accuracy: ' + str(accuracy_score(Y_test, Y_pred)))
print('F-score: ' + str(f1_score(Y_test, Y_pred)))
```

```
Accuracy: 0.7196969696969697
F-score: 0.5316455696202531
```

In [36]:

```
#Getting the coefficients
c=list(model.coef_[0])
labels=list(X_train.columns)

#Creating a dataframe that stores the coefficients
f=pd.DataFrame()
f['Features'] = labels
f['importance'] = c

f.sort_values(by=['importance'], ascending=True, inplace=True)
f['positive'] = f['importance'] > 0
f.set_index('Features', inplace=True)

#Plotting the graph for Feature Importance
f.importance.plot(kind='barh', figsize=(11, 6),color = f.positive.map({True: 'blue', Fal
plt.title('Feature Importance')
plt.xlabel('Importance')
```

Out[36]: Text(0.5, 0, 'Importance')



In [37]:
```
#Training the model with the new set of variables
col=['Glucose','Pregnancies','BMI','DiabetesPedigree','SkinThickness','Age','Insulin']
X=df[col]
Y=df['Outcome']

#Training the Logistic regression model
model=LogisticRegression()
model.fit(X,Y)

#Predicting the Outcome based on the model trained
test['Outcome']=model.predict(test[['Glucose','Pregnancies','BMI','DiabetesPedigree','Sk
test.head(10)
```

Out[37]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigree	Age	Outco
0	0.287183	-0.156221	-0.595854	0.000000	0.000000	-0.522552	1.949902	-0.885186	
1	-0.789754	-0.742052	0.320844	0.631103	-1.581139	0.759179	-0.558983	-0.147531	
2	-0.071796	-1.249771	-1.512553	-1.596319	0.000000	-1.377038	-0.662498	-0.516359	
3	-1.148733	1.601269	1.466718	1.373577	1.581139	1.465774	-0.665222	-0.393416	
4	1.723100	0.546775	0.320844	-0.408361	0.000000	-0.325362	-0.063199	1.942492	

In [38]:
```
#Finding the probability with Outcome 1
p=model.predict_proba(test[['Glucose','Pregnancies','BMI','DiabetesPedigree','SkinThickn

prob_diabetes = p.mean()

print("Probability of getting diabetes: " + str(prob_diabetes))
```

```
Probability of getting diabetes: 0.38903014688944976
```