

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

Assignment 1 Diabetes DATA analysis

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

Dataset description

```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI
0	NaN	148.0	30	35.0	NaN	33.6
1	1.0	85.0	66	29.0	0.0	26.6
2	8.0	183.0	64	0.0	0.0	23.3
3	1.0	89.0	66	23.0	94.0	28.1
4	0.0	137.0	40	35.0	168.0	43.1

	DiabetesPedigreeFunction	Age	Outcome
0	2.000	50.0	1.0
1	0.351	31.0	0.0
2	0.672	32.0	1.0
3	0.167	21.0	0.0
4	2.288	33.0	1.0

Getting features type

```
df.dtypes
```

```
Pregnancies    float64
Glucose         float64
BloodPressure   int64
SkinThickness   float64
Insulin         float64
BMI            float64
```

```
DiabetesPedigreeFunction    float64
Age                        float64
Outcome                    float64
dtype: object
```

Print columns

```
df.columns

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

Getting the count of missing values

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

Pregnancies    5
Glucose         2
BloodPressure   0
SkinThickness   2
Insulin         3
BMI             2
DiabetesPedigreeFunction  3
Age             2
Outcome         1
dtype: int64
```

Deleting columns that have 1% missing values

```
missing_percentage = df.isnull().mean() * 100

threshold = 1

# Filter columns with more than the threshold percentage of missing values

columns_to_drop = missing_percentage[missing_percentage >
threshold].index

# Drop the columns

df = df.drop(columns=columns_to_drop)

# Display the resulting dataframe after removing columns with too many missing values

print(f"Columns dropped: {list(columns_to_drop)}")

Columns dropped: []
```

```

print(f"Remaining columns: {df.columns}")

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

## Missing values print

print(df.isnull().sum())

Pregnancies      5
Glucose           2
BloodPressure     0
SkinThickness     2
Insulin           3
BMI               2
DiabetesPedigreeFunction  3
Age               2
Outcome           1
dtype: int64

## Delete Missing Values Rows

df = df.dropna()

print(df.isnull().sum())

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

```

Pre-processing on dataset

```

from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()

for column in df.select_dtypes(include=['object']).columns: df[column]
= label_encoder.fit_transform(df[column])

df.head()

Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin
BMI  \

```

1	1.0	85.0	66	29.0	0.0	26.6
2	8.0	183.0	64	0.0	0.0	23.3
3	1.0	89.0	66	23.0	94.0	28.1
4	0.0	137.0	40	35.0	168.0	43.1
5	5.0	116.0	74	0.0	0.0	25.6

	DiabetesPedigreeFunction	Age	Outcome
1	0.351	31.0	0.0
2	0.672	32.0	1.0
3	0.167	21.0	0.0
4	2.288	33.0	1.0
5	3.000	30.0	0.0

Filling missing values based on feature type

```
# Fill missing values in columns with mode for column in
df.select_dtypes(include=['float64']).columns: mean_value =
df[column].mean() # Get the mode (most frequent value) df[column] =
df[column].fillna(mean_value)
# Fill missing values in categorical columns with mode for column in
df.select_dtypes(include=['object']).columns: mode_value =
df[column].mode()[0] # Get the mode (most frequent value) df[column] =
df[column].fillna(mode_value)
df.isnull().sum()
```

Pregnancies	0
Glucose	0
BloodPressure	0
SkinThickness	0
Insulin	0
BMI	0
DiabetesPedigreeFunction	0
Age	0
Outcome	0

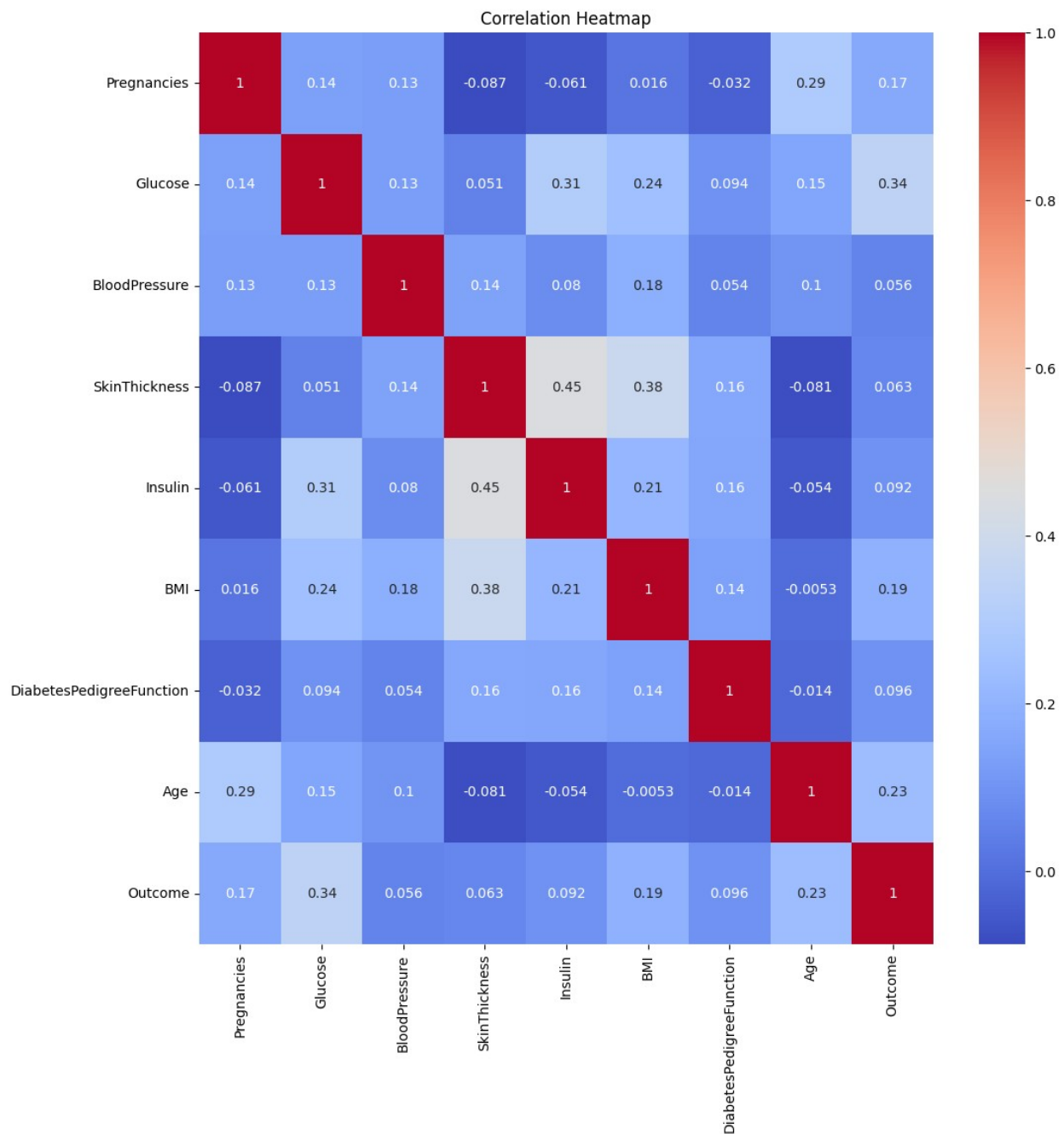
dtype: int64

Data Visualization

Correlation heatmap

```
import seaborn as sns
plt.figure(figsize=(12, 12))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
```

```
plt.title('Correlation Heatmap')
plt.show()
```



Diabetes Distribution

```
import matplotlib.pyplot as plt

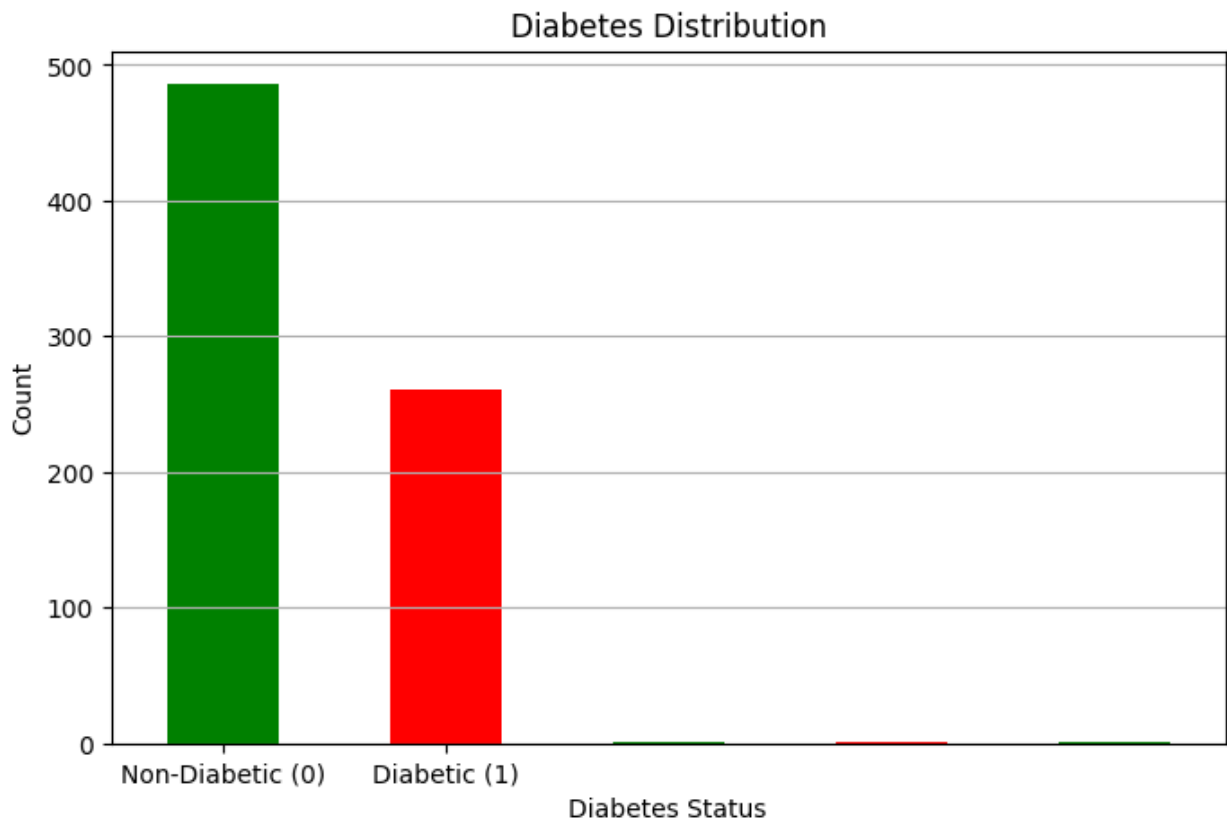
plt.figure(figsize=(8, 5))
df['Outcome'].value_counts().sort_index().plot(kind='bar',
```

```

color=['green', 'red'])

plt.xticks([0, 1], ['Non-Diabetic (0)', 'Diabetic (1)'], rotation=0)
plt.title('Diabetes Distribution')
plt.xlabel('Diabetes Status')
plt.ylabel('Count')
plt.grid(axis='y')
plt.show()

```



```

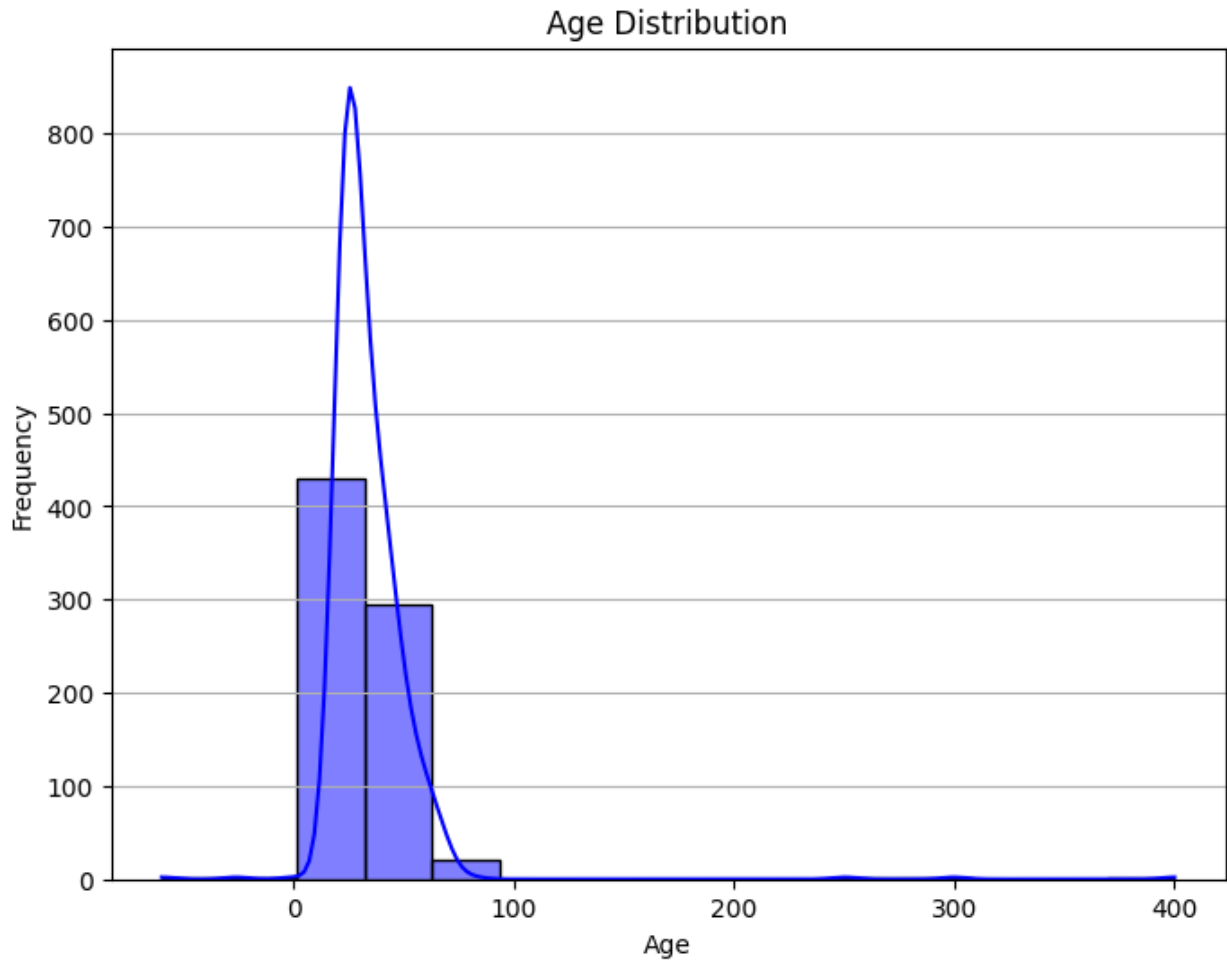
## Age Distribution

import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
sns.histplot(df['Age'], bins=15, kde=True, color='blue')

plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid(axis='y')
plt.show()

```



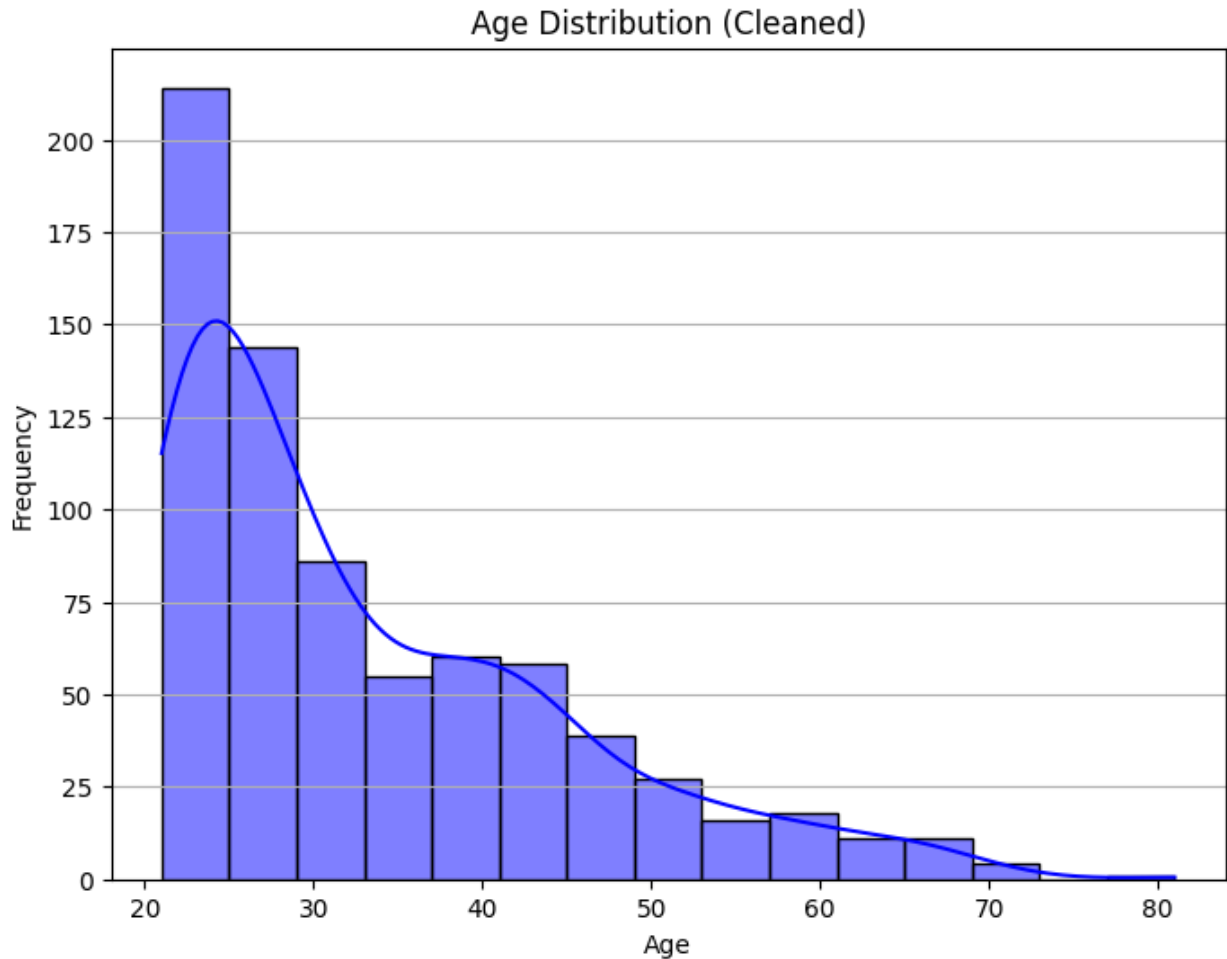
```
# Data outliers handle

df_clean = df[(df['Age'] >= 1) & (df['Age'] <= 100)]

import seaborn as sns
import matplotlib.pyplot as plt

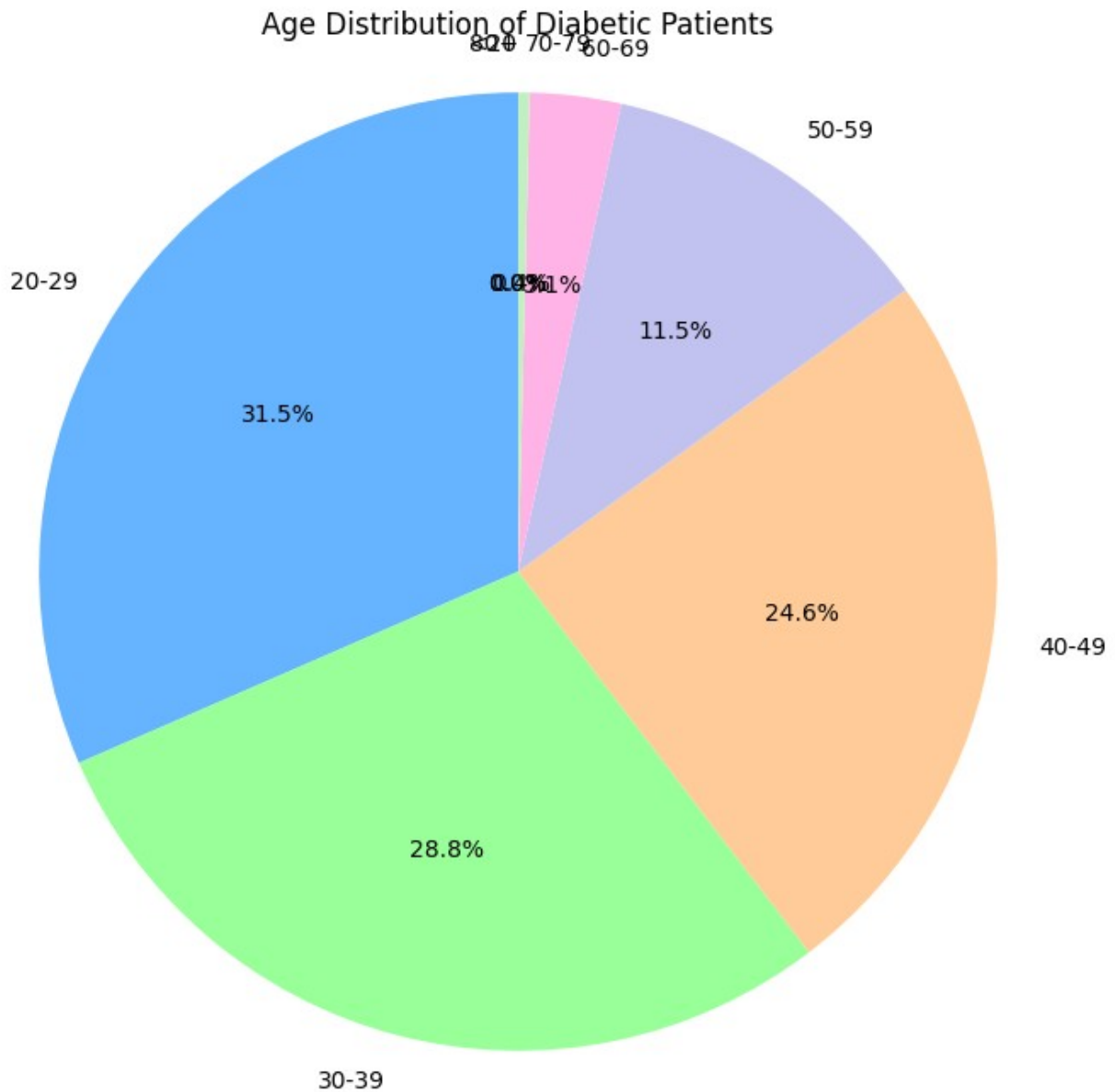
plt.figure(figsize=(8, 6))
sns.histplot(df_clean['Age'], bins=15, kde=True, color='blue')

plt.title('Age Distribution (Cleaned)')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid(axis='y')
plt.show()
```



```
import matplotlib.pyplot as plt
import pandas as pd
bins = [0, 20, 30, 40, 50, 60, 70, 80, 100]
labels = ['<20', '20-29', '30-39', '40-49', '50-59', '60-69', '70-79',
'80+']
df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels,
right=False)

diabetic = df[df['Outcome'] == 1]
age_counts = diabetic['AgeGroup'].value_counts().sort_index()
colors =
['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0', '#ffb3e6', '#c2f0c2',
'#f0c2c2']
plt.figure(figsize=(8, 8))
plt.pie(age_counts, labels=age_counts.index, autopct='%1.1f%%',
startangle=90, colors=colors)
plt.title('Age Distribution of Diabetic Patients')
plt.axis('equal')
plt.show()
```

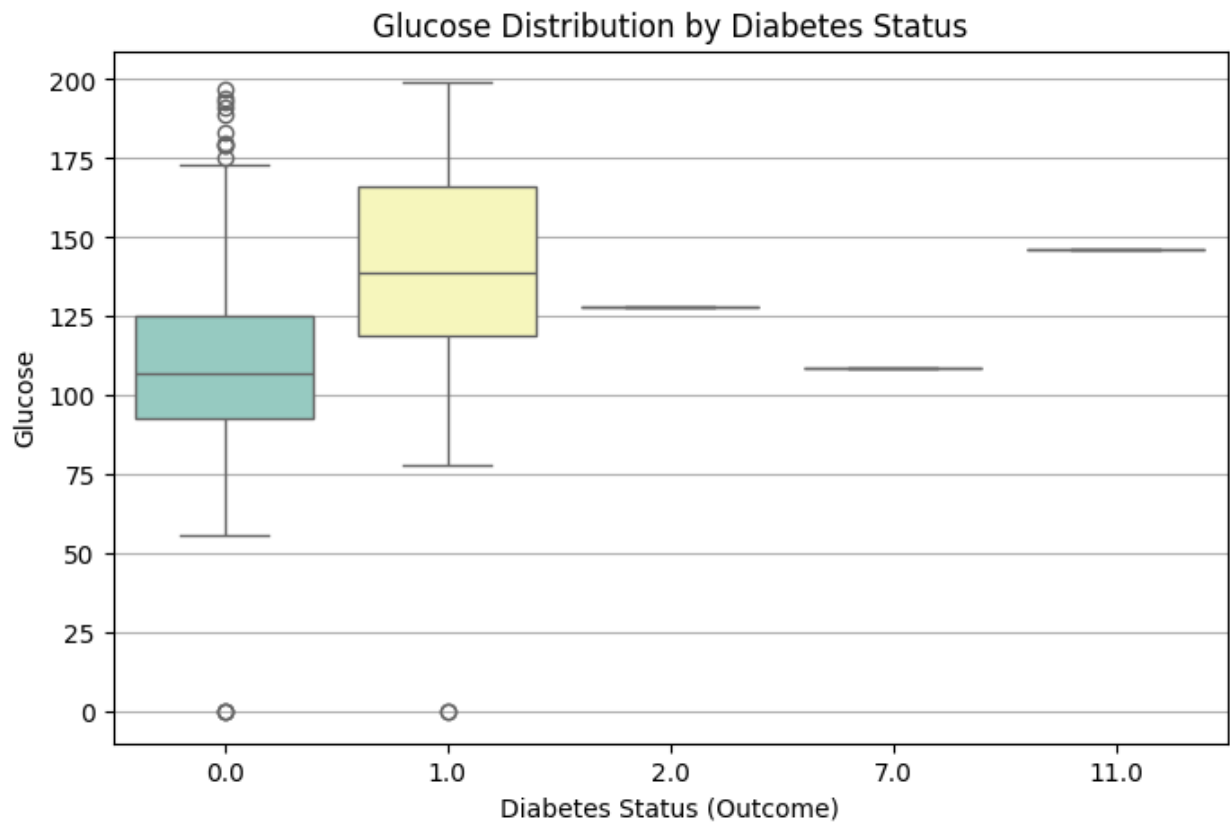


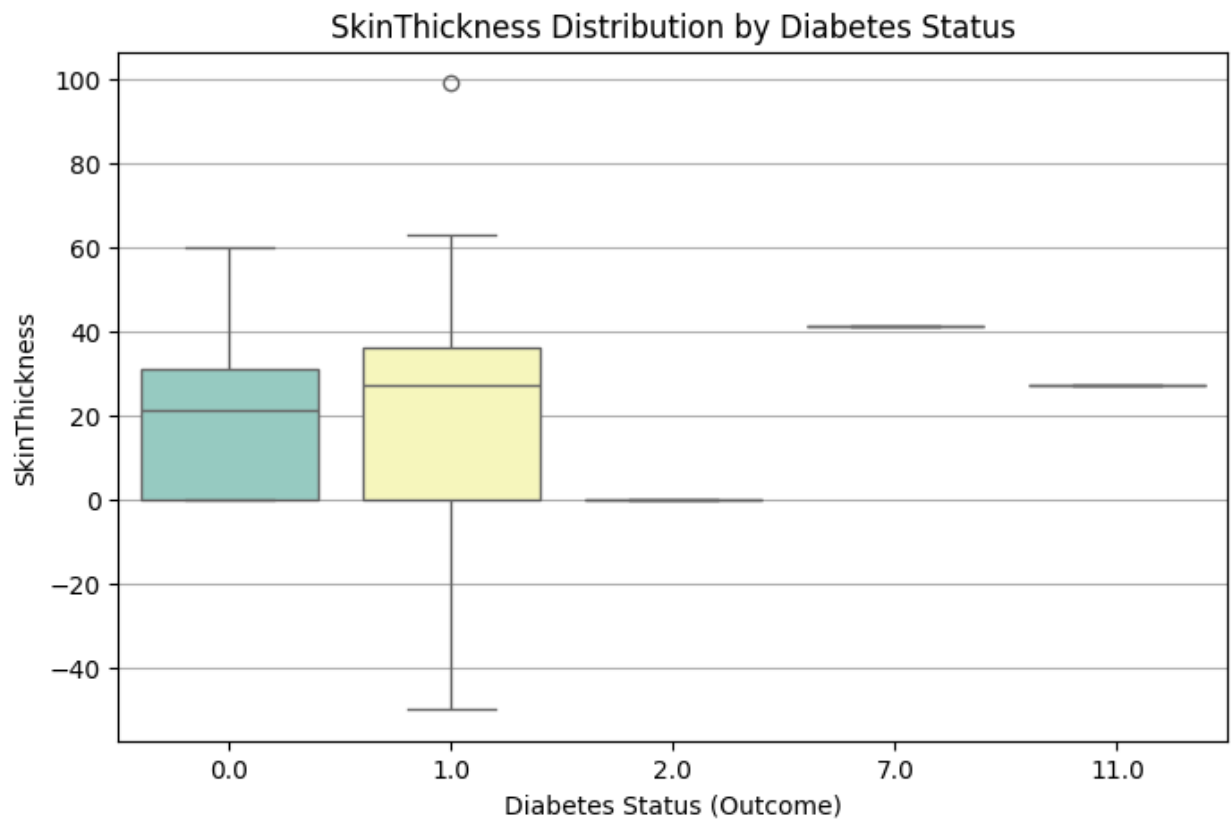
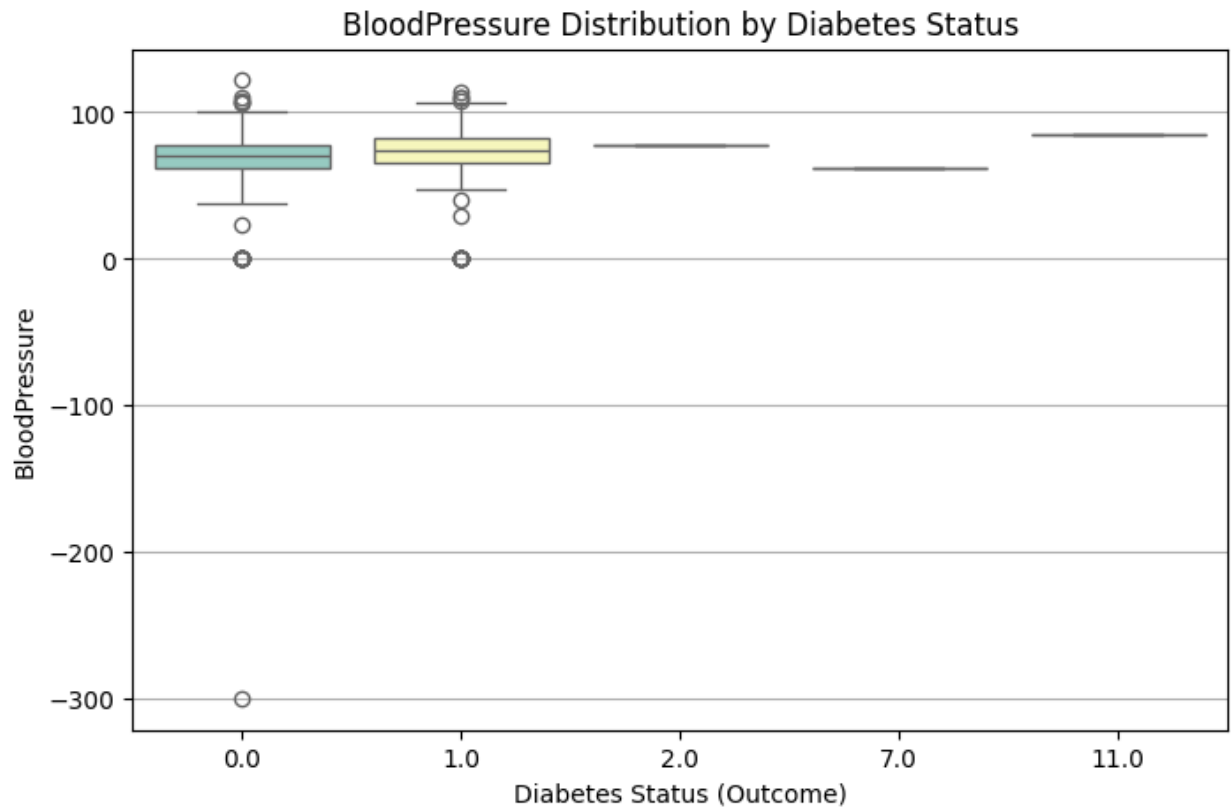
```
import seaborn as sns
import matplotlib.pyplot as plt

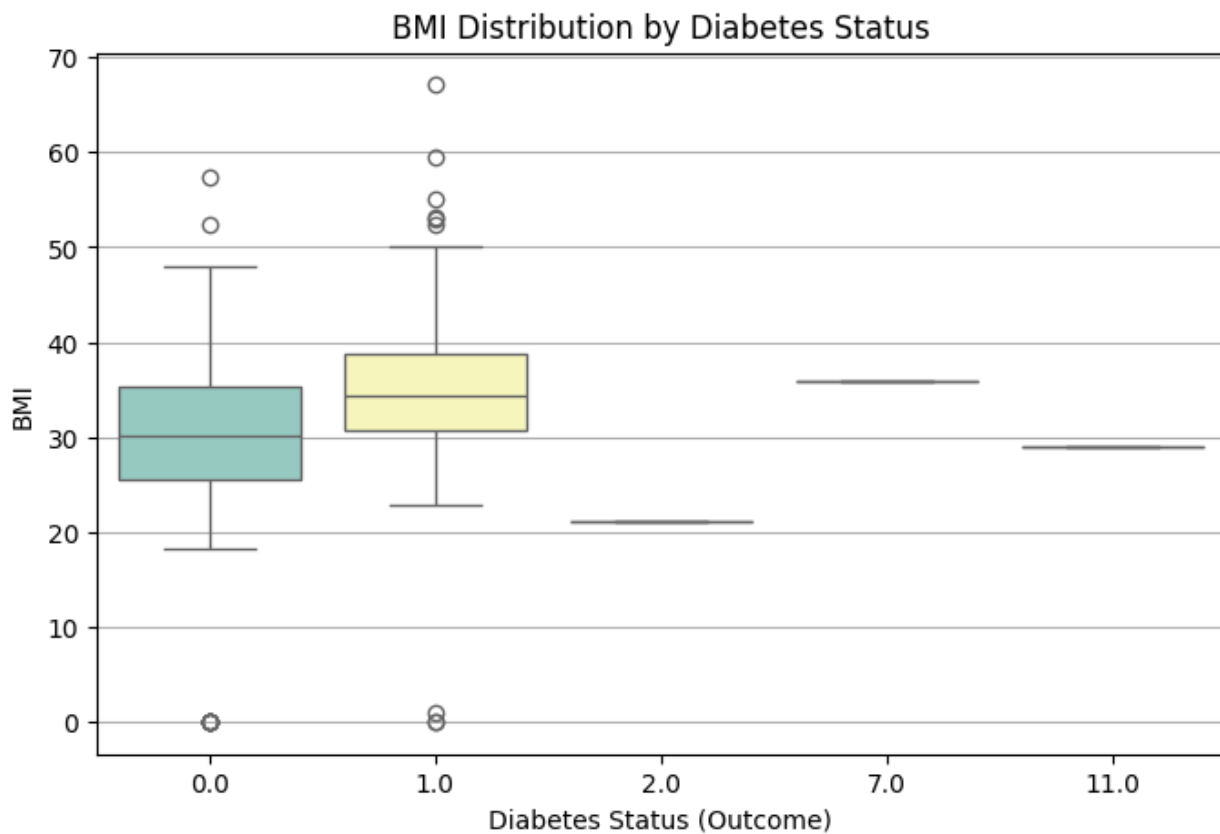
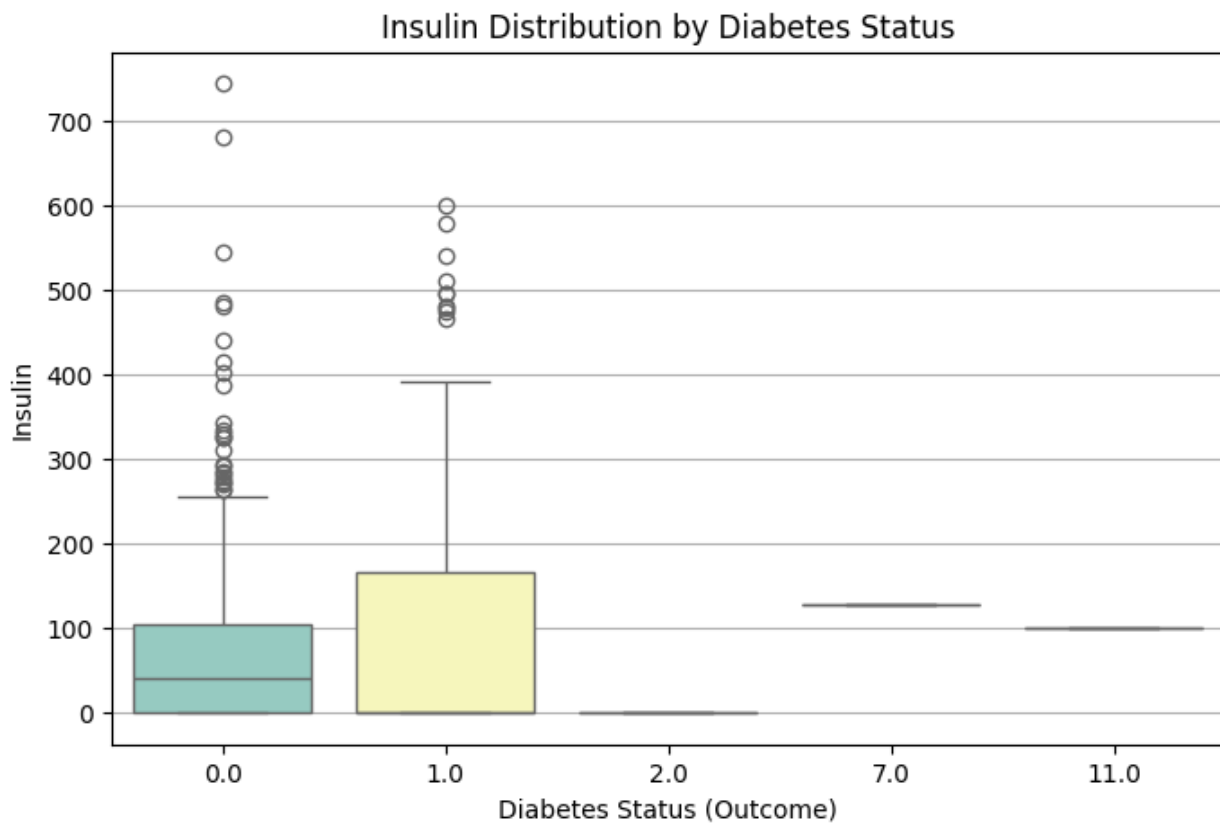
features = ['Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
            'BMI', 'DiabetesPedigreeFunction', 'Age']

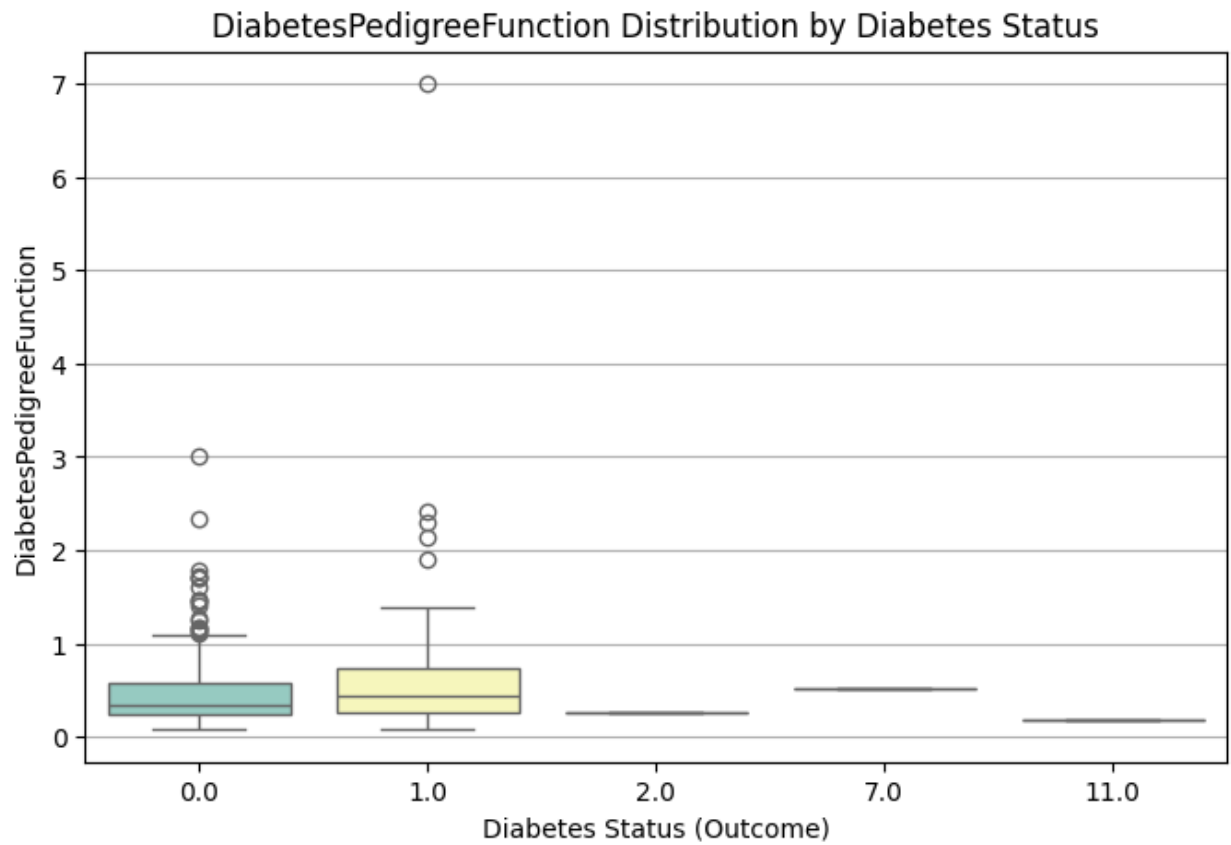
for feature in features:
    plt.figure(figsize=(8, 5))
    sns.boxplot(x='Outcome', y=feature, data=df, palette='Set3')
    plt.title(f'{feature} Distribution by Diabetes Status')
    plt.xlabel('Diabetes Status (Outcome)')
    plt.ylabel(feature)
```

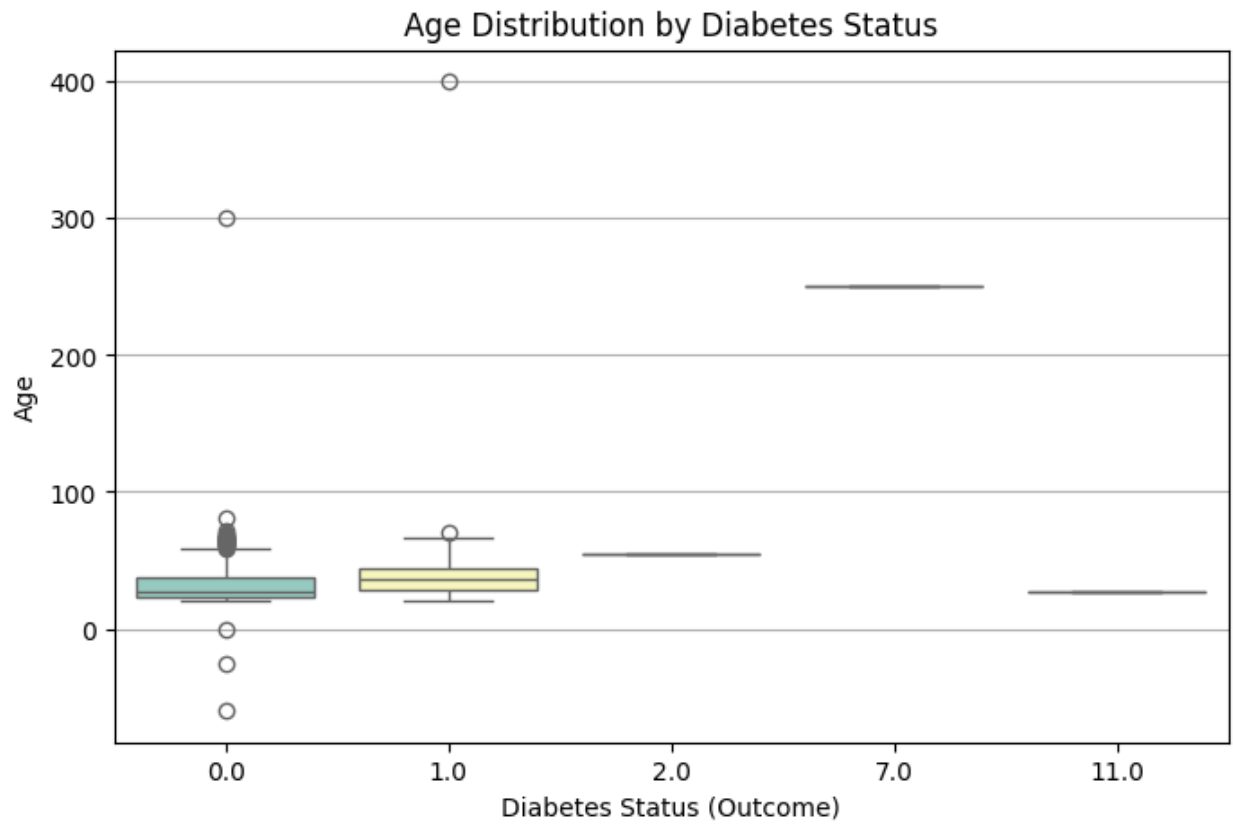
```
plt.grid(axis='y')  
plt.show()
```











BR,