Exploring the Dynamics of Diabetes: A Comprehensive Synthetic Dataset Analysis

Name: Vinayak V Thayil

Roll No: AM.EN.U4CSE21161

```
!pip install faker
```

```
Requirement already satisfied: faker in /usr/local/lib/python3.10/dist-packages
(22.0.0)
Requirement already satisfied: python-dateutil>=2.4 in
/usr/local/lib/python3.10/dist-packages (from faker) (2.8.2)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.4->faker)
(1.16.0)
```

```
from faker import Faker
import pandas as pd
import random
import numpy as np
from tabulate import tabulate
# Initialize Faker
fake = Faker()

# Set seed for reproducibility
random.seed(42)

# Generate synthetic data with missing values
num_records = 2000

data = {
    'Patient_ID': [fake.random_int(min=10000, max=99999999) for _ in
    range(num_records)],
    'Age': [random.randint(18, 85) for _ in range(num_records)],
    'Gender': [fake.random_element(['Male', 'Female']) for _ in
    range(num_records)],
```

```
    'BMI': [random.uniform(18.5, 40.0) for _ in range(num_records)],
    'Blood_Pressure': [f"{random.randint(90, 160)}/{random.randint(60,
    100)}" for _ in range(num_records)],
    'Glucose_Level': [random.randint(70, 200) if random.random() > 0.1 else
    np.nan for _ in range(num_records)],
    'Cholesterol_Level': [fake.random_element(['Normal', 'High']) if
    random.random() > 0.05 else np.nan for _ in range(num_records)],
    'Physical_Activity': [random.uniform(0, 10) for _ in
    range(num_records)],
    'Diet_Type': [fake.random_element(['Balanced', 'High Carb', 'High
    Protein']) for _ in range(num_records)],
    'Insulin_Usage': [fake.random_element(['Yes', 'No']) for _ in
    range(num_records)],
    'HbA1c_Level': [random.uniform(4.0, 10.0) if random.random() > 0.1 else
    np.nan for _ in range(num_records)]
}

# Convert the dictionary to a DataFrame
df = pd.DataFrame(data)

# Save the synthetic dataset with missing values to a CSV file
df.to_csv('Diabetes.csv', index=False)
```

```
df.shape
```

```
(2000, 11)
```

```
# Display dataset in a table
print(tabulate(df.head(), headers='keys', tablefmt='fancy_grid'))
```

```
       Patient_ID     Age   Gender         BMI   Blood_Pressure
Glucose_Level   Cholesterol_Level      Physical_Activity   Diet_Type
Insulin_Usage       HbA1c_Level

  0       89969230      32   Male       33.1274   135/62
164   High                              4.92227   High Protein   Yes
5.29544

  1       58207332      21   Male       31.31     118/91
183   Normal                            6.69642   Balanced       No
9.96815

  2       68479409      53   Male       35.4285   119/64
nan   Normal                            6.97125   Balanced       Yes
7.64039
```

```
  3      24034624     49   Male      29.7984   123/83
141   Normal                          5.50031   Balanced      No
7.97526

  4      19914407     46   Male      29.8557   119/62
161   Normal                          0.4013    High Protein  Yes
7.90114
```

```
# Display  dataset in a table
print(tabulate(df.tail(), headers='keys', tablefmt='fancy_grid'))
```

```
         Patient_ID     Age   Gender         BMI   Blood_Pressure
Glucose_Level   Cholesterol_Level       Physical_Activity   Diet_Type
Insulin_Usage       HbA1c_Level

 1995      99462719     65   Female     30.6767   93/88
104   Normal                          6.40239   Balanced      Yes
9.88242

 1996      49654318     71   Male       30.665    105/82
153   Normal                          4.57181   High Carb     No
4.22461

 1997      40710714     78   Female     37.2522   138/98
178   High                            4.17704   Balanced      No
7.20969

 1998      76210876     45   Male       21.6433   135/84
88   nan                             8.91607   High Protein  Yes
9.48165

 1999      76392323     49   Female     34.2381   112/60
76   Normal                          6.41936   Balanced      No
nan
```

```
df.dtypes
```

```
Patient_ID             int64
Age                    int64
Gender                object
BMI                  float64
Blood_Pressure        object
```

```
Glucose_Level        float64
Cholesterol_Level    float64
Physical_Activity    float64
Diet_Type             object
Insulin_Usage         object
HbA1c_Level          float64
dtype: object
```

```
#summary statistics of the DataFrame
summary_stats_selected = tabulate(df.describe(), headers='keys',
tablefmt='fancy_grid')
print(summary_stats_selected)
```

```
             Patient_ID         Age          BMI     Glucose_Level
Physical_Activity     HbA1c_Level

  count    2000             2000        2000               1789              2000
     1770

  mean        5.02892e+07     52.0685     29.3303           135.224
4.9823              7.05503

  std         2.86118e+07     19.3418      6.15315           38.1555
2.90219             1.73724

  min     75089               18          18.5387           70
9.4794e-05          4.0039

  25%         2.51296e+07     35          23.9095           102
2.45643             5.49241

  50%         5.09571e+07     53          29.6082           134
5.01912             7.1125

  75%         7.47029e+07     69          34.5575           169
7.5596              8.57603

  max         9.99368e+07     85          39.9877           200
9.98145             9.99066
```

```
df.nunique()
```

```
Patient_ID           2000
Age                    68
```

```
Gender                  2
BMI                  2000
Blood_Pressure       1449
Glucose_Level         131
Cholesterol_Level       2
Physical_Activity    2000
Diet_Type               3
Insulin_Usage           2
HbA1c_Level          1770
dtype: int64
```

```
#missing Values
missing_values = df.isnull().sum()
print("Missing Values:")
print(missing_values)
```

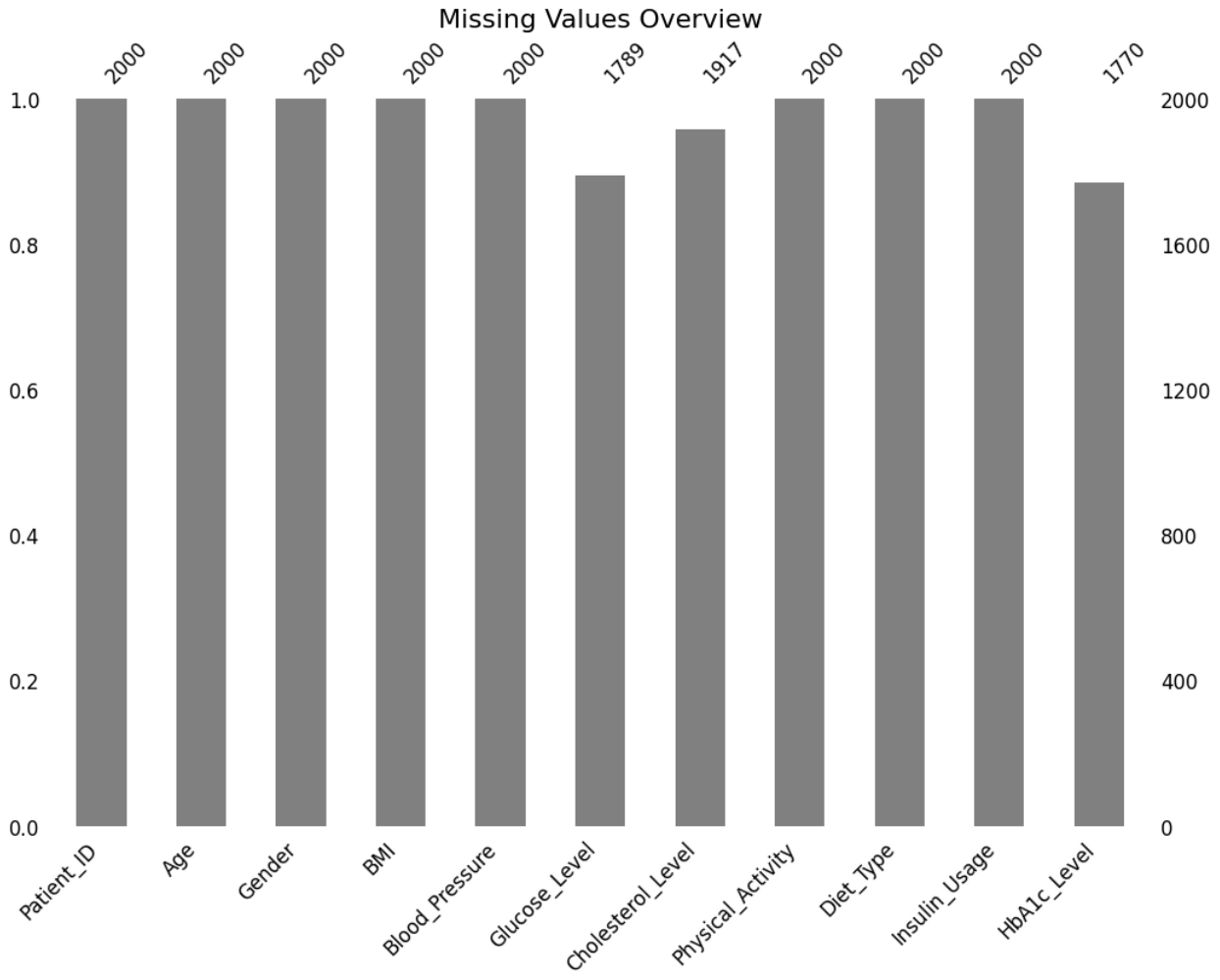
```
Missing Values:
Patient_ID             0
Age                    0
Gender                 0
BMI                    0
Blood_Pressure         0
Glucose_Level        211
Cholesterol_Level     83
Physical_Activity      0
Diet_Type              0
Insulin_Usage          0
HbA1c_Level          230
dtype: int64
```

```
# Display the number of missing values before removal
print("Number of missing values before removal:")
print(df.isnull().sum())

# Remove rows with missing values
df_cleaned = df.dropna()

# Display the number of missing values after removal
print("\nNumber of missing values after removal:")
print(df_cleaned.isnull().sum())
```

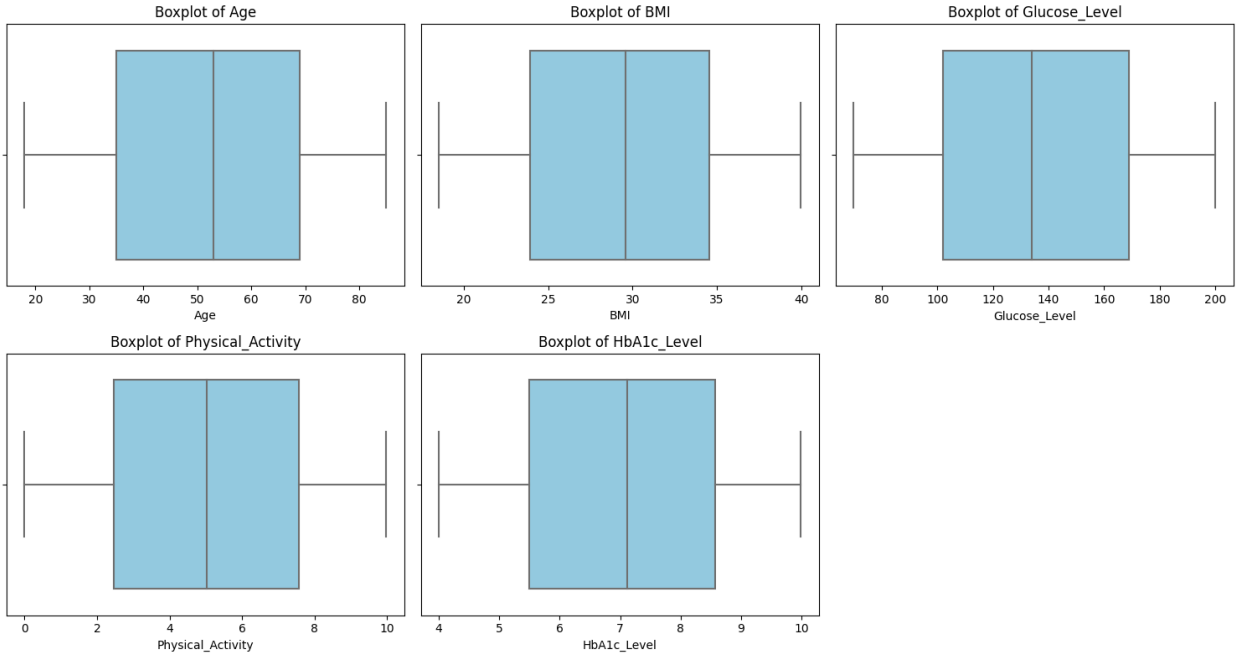
```
Number of missing values before removal:
Patient_ID             0
Age                    0
Gender                 0
```

```
BMI                   0
Blood_Pressure        0
Glucose_Level       211
Cholesterol_Level    83
Physical_Activity     0
Diet_Type             0
Insulin_Usage         0
HbA1c_Level         230
dtype: int64

Number of missing values after removal:
Patient_ID          0
Age                 0
Gender              0
BMI                 0
Blood_Pressure      0
Glucose_Level       0
Cholesterol_Level   0
Physical_Activity   0
Diet_Type           0
Insulin_Usage       0
HbA1c_Level         0
dtype: int64
```

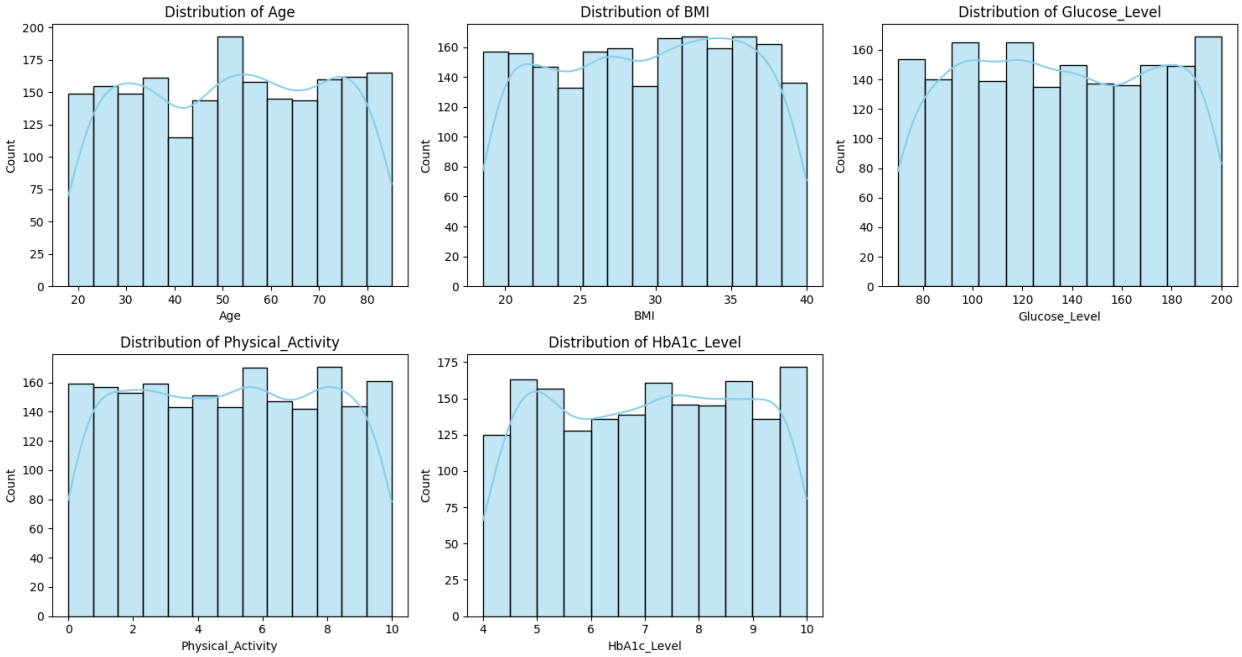
```python
import missingno as msno
import matplotlib.pyplot as plt
# Plot the missing values using missingno.bar
msno.bar(df, figsize=(12, 8), color='grey', fontsize=12, labels=True)
plt.title('Missing Values Overview', fontsize=16)
plt.show()
```



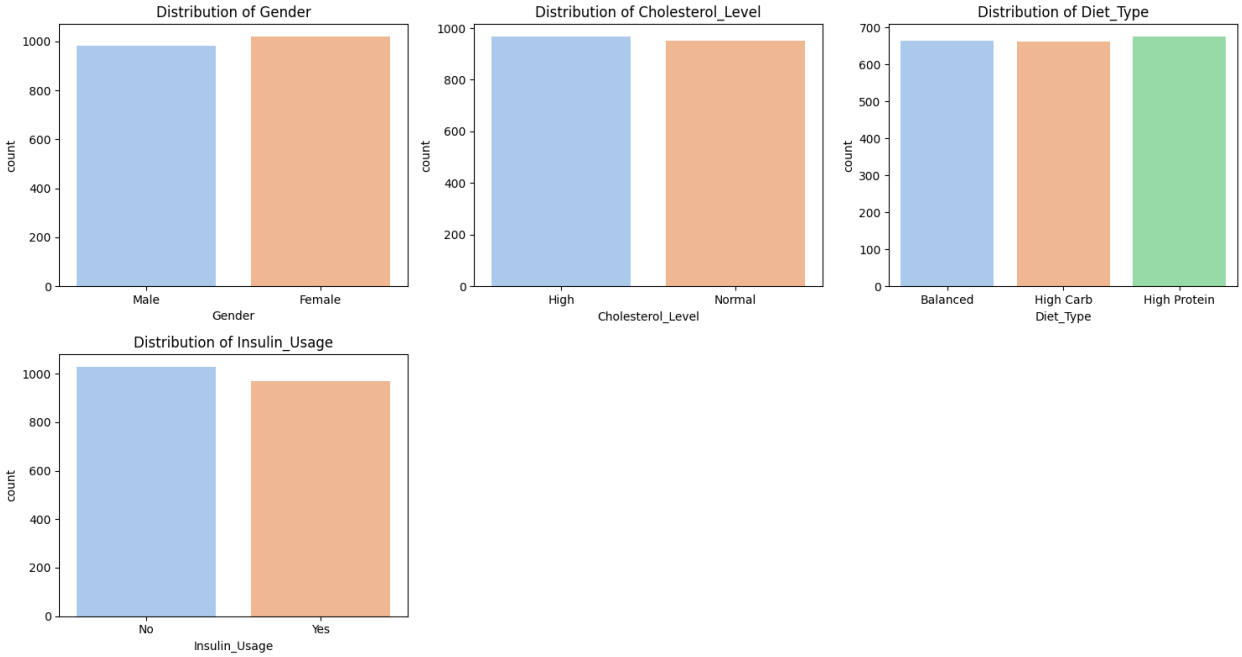

```
import seaborn as sns
import matplotlib.pyplot as plt
numerical_columns = ['Age', 'BMI', 'Glucose_Level', 'Physical_Activity',
'HbA1c_Level']
# Create box plots for numerical columns
plt.figure(figsize=(15, 8))
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(x=df[column], color='skyblue')
    plt.title(f'Boxplot of {column}')

plt.tight_layout()
plt.show()
```



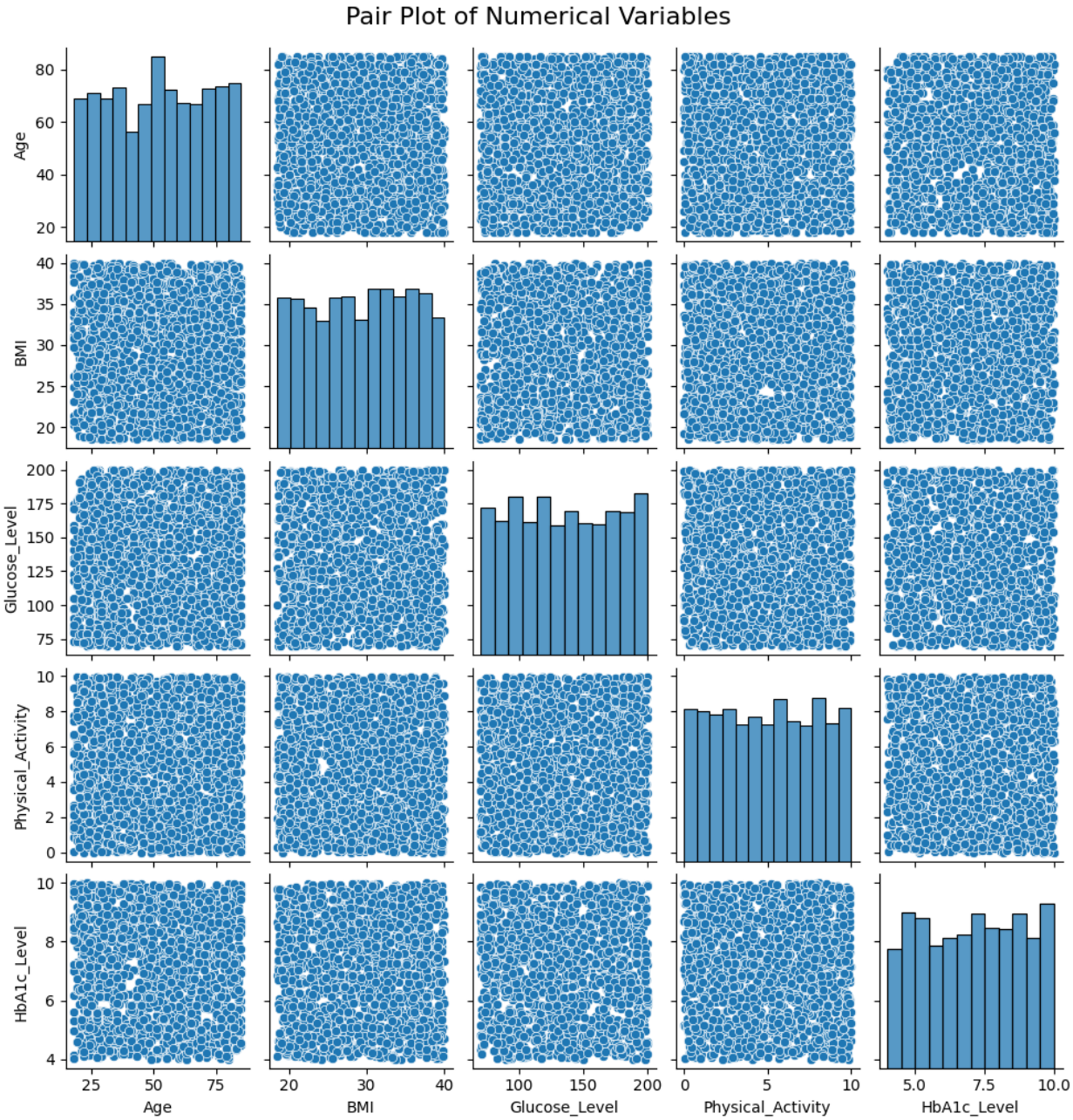

```
numerical_columns = ['Age', 'BMI', 'Glucose_Level', 'Physical_Activity',
'HbA1c_Level']

#
plt.figure(figsize=(15, 8))
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    sns.histplot(df[column], kde=True, color='skyblue')
    plt.title(f'Distribution of {column}')

plt.tight_layout()
plt.show()
```




```
categorical_columns = ['Gender', 'Cholesterol_Level','Diet_Type',
'Insulin_Usage']

# Plot bar plots for categorical columns
plt.figure(figsize=(15, 8))
for i, column in enumerate(categorical_columns, 1):
    plt.subplot(2, 3, i)
    sns.countplot(x=df[column], palette='pastel')
    plt.title(f'Distribution of {column}')

plt.tight_layout()
plt.show()
```




```
numerical_columns = ['Age', 'BMI', 'Glucose_Level', 'Physical_Activity',
'HbA1c_Level']

# Create a pair plot using Seaborn
sns.pairplot(df[numerical_columns], height=2)
plt.suptitle('Pair Plot of Numerical Variables', y=1.02, fontsize=16)
plt.show()
```

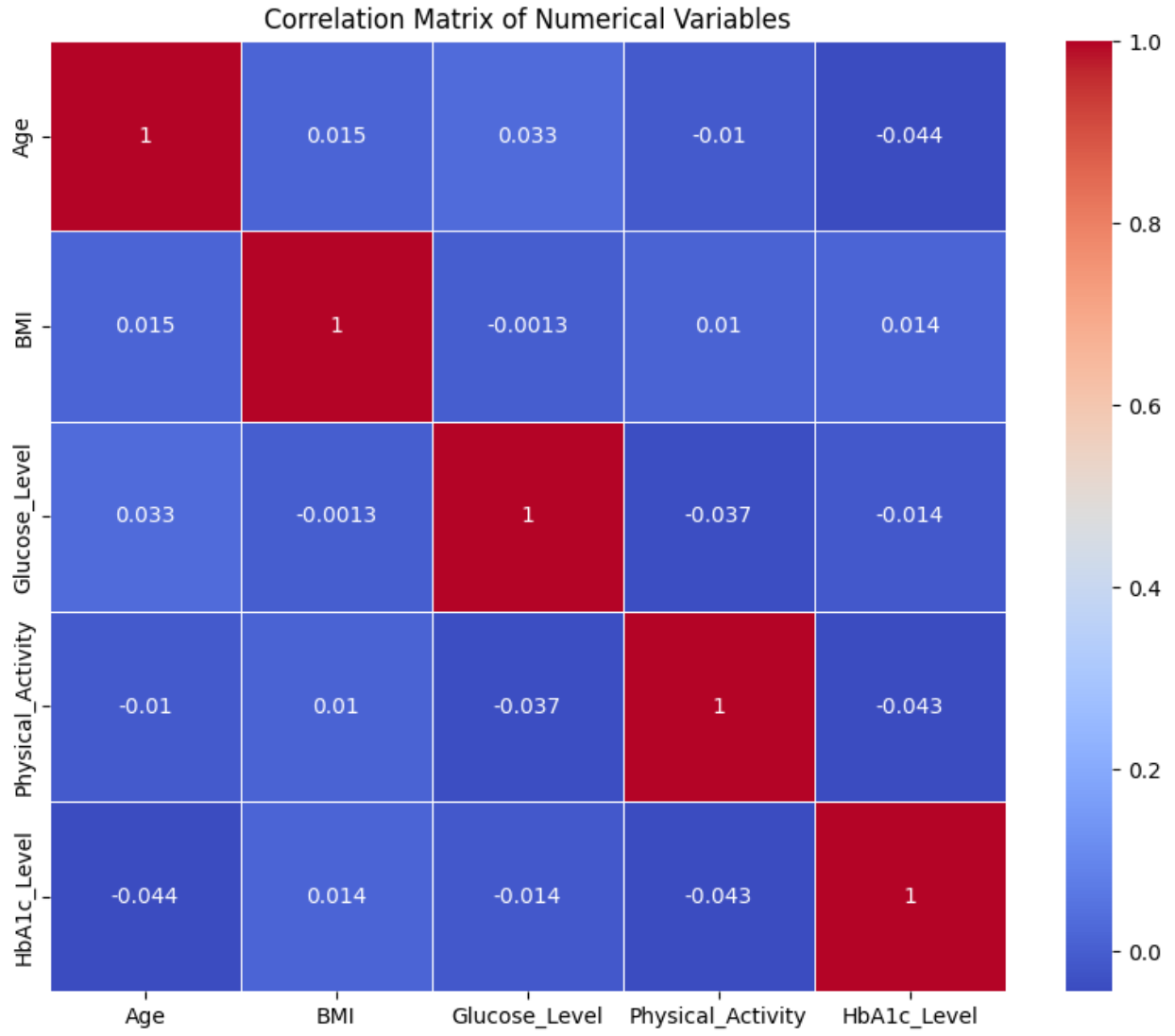



```
# Calculate the correlation matrix
correlation_matrix = tabulate(df[numerical_columns].corr(),headers='keys',
tablefmt='fancy_grid')
print(correlation_matrix)
```

```
                          Age          BMI     Glucose_Level
Physical_Activity     HbA1c_Level
```

```
 Age                  1            0.014684          0.0326281
-0.0103207      -0.044441

 BMI                  0.014684     1                -0.00132464
0.0099931       0.0144518

 Glucose_Level        0.0326281   -0.00132464        1
-0.0372564      -0.0137819

 Physical_Activity   -0.0103207    0.0099931        -0.0372564              1
    -0.0434523

 HbA1c_Level         -0.044441     0.0144518        -0.0137819
-0.0434523       1
```

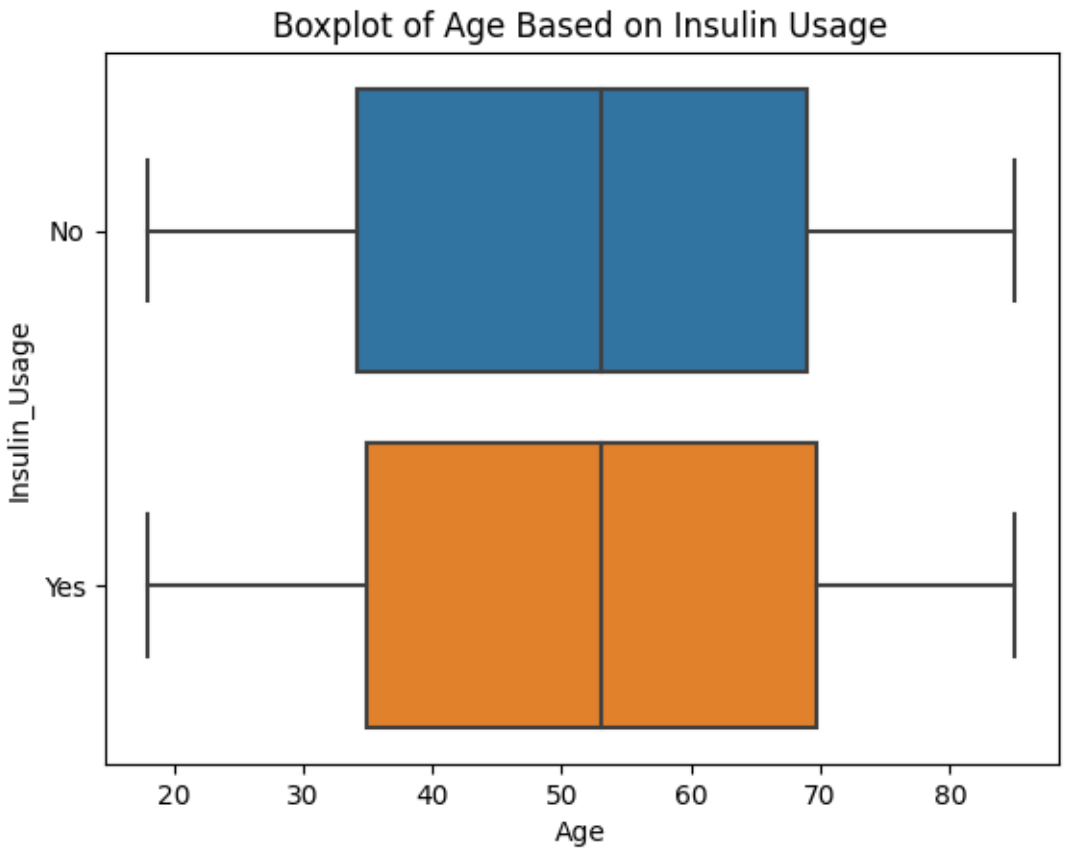
```python
# Create a heatmap of the correlation matrix
correlation_matrix = df[numerical_columns].corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=.5)
plt.title('Correlation Matrix of Numerical Variables')
plt.show()
```



Q1.How does the distribution of ages vary based on insulin usage in the given diabetes dataset? * Observe if the medians of the two groups (Insulin Usage and No Insulin Usage) differ significantly. A substantial difference might indicate an association between age and insulin usage. * Check for differences in the spread of ages within each group. A wider spread in one group might suggest greater age variability.

```
sns.boxplot(x='Age',y='Insulin_Usage',data=df)
plt.title('Boxplot of Age Based on Insulin Usage')
```

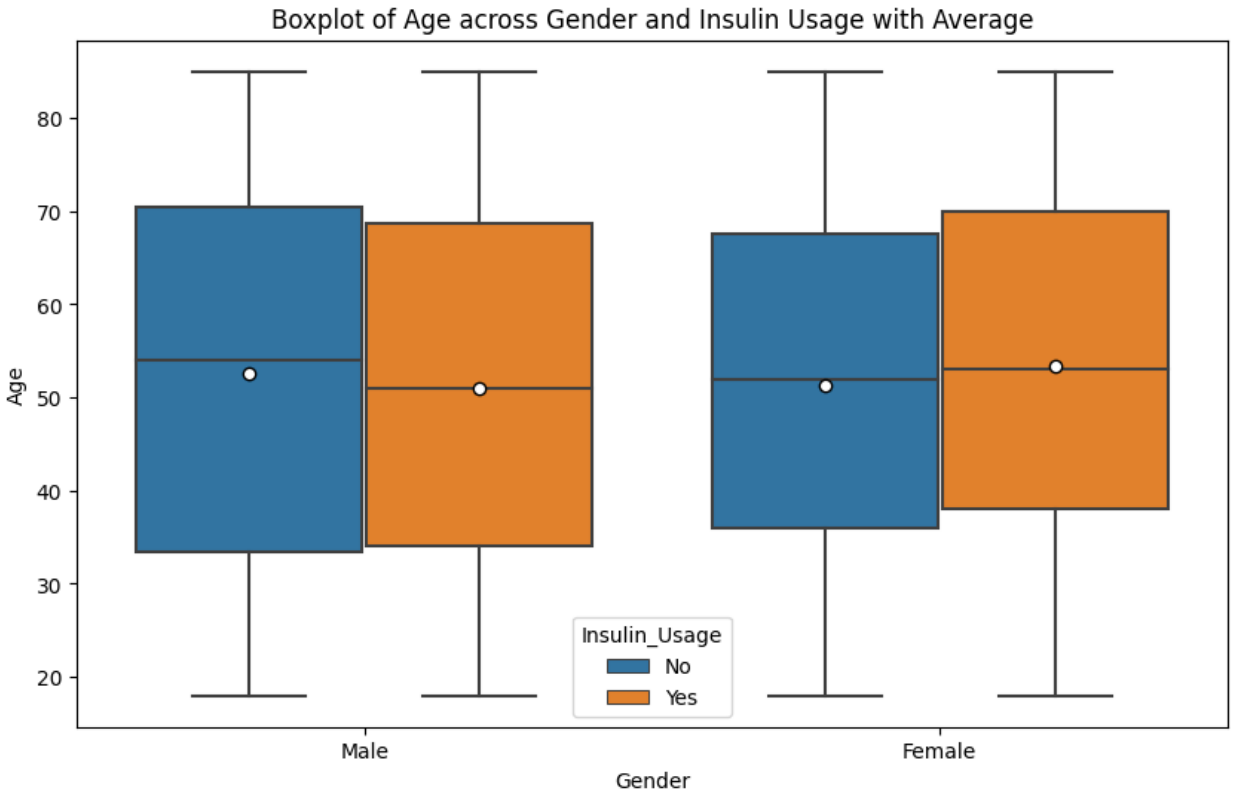
```
Text(0.5, 1.0, 'Boxplot of Age Based on Insulin Usage')
```



Q2. How does the distribution of ages vary among different genders, considering the use of insulin?

Boxplot enables a comprehensive exploration of age distributions, gender differences, and the impact of insulin usage within the diabetes dataset. Further statistical analysis or subgroup comparisons may provide deeper insights into these observed patterns.

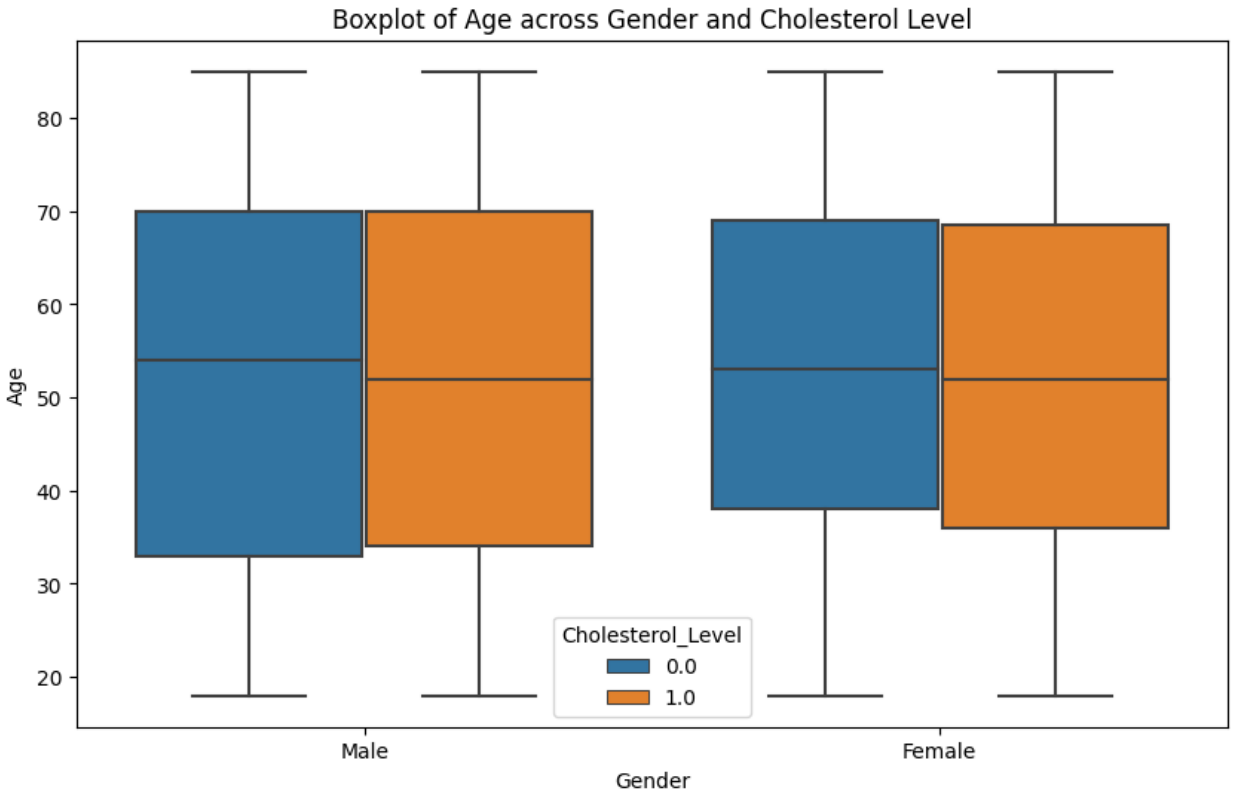
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='Age', hue='Insulin_Usage', data=df,
showmeans=True, meanprops={"marker":"o", "markerfacecolor":"white",
"markeredgecolor":"black"})
plt.title('Boxplot of Age across Gender and Insulin Usage with Average')
plt.show()
```



Q3. How does age vary across different genders concerning cholesterol levels?

The boxplot visualization reveals interesting insights into the distribution of age across different genders, considering varying levels of cholesterol within the diabetes dataset. By examining the boxes and whiskers, we can discern the central tendency, spread, and potential outliers in the data.

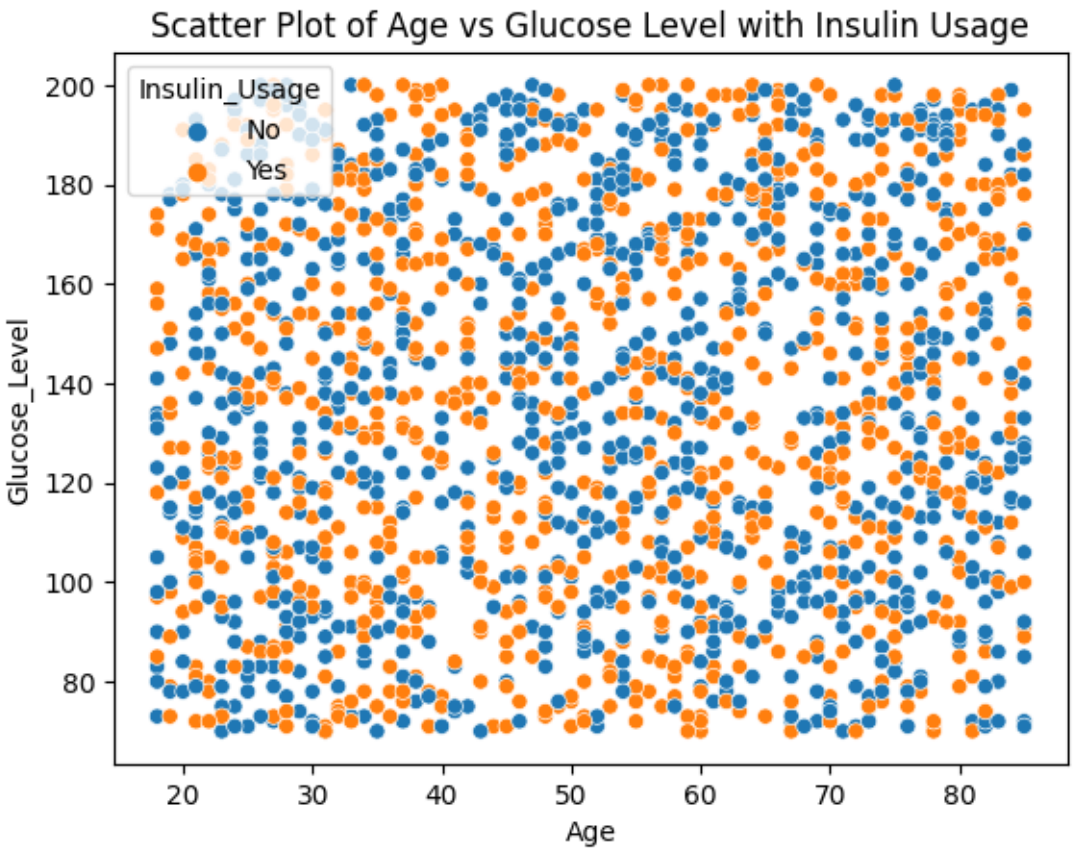
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='Gender', y='Age', hue='Cholesterol_Level', data=df)
plt.title('Boxplot of Age across Gender and Cholesterol Level')
plt.show()
```



Q1.How does the scatter plot of age versus glucose level with insulin usage differentiate patterns in the dataset?

The scatter plot of age versus glucose level with insulin usage provides a visual representation of the dataset's dynamics, allowing us to discern patterns and potential correlations. By incorporating insulin usage as a hue factor, the plot enables a more nuanced exploration of how this variable influences the relationship between age and glucose levels.

```
sns.scatterplot(x='Age', y='Glucose_Level', hue='Insulin_Usage', data=df)
plt.title('Scatter Plot of Age vs Glucose Level with Insulin Usage')
plt.show()
```

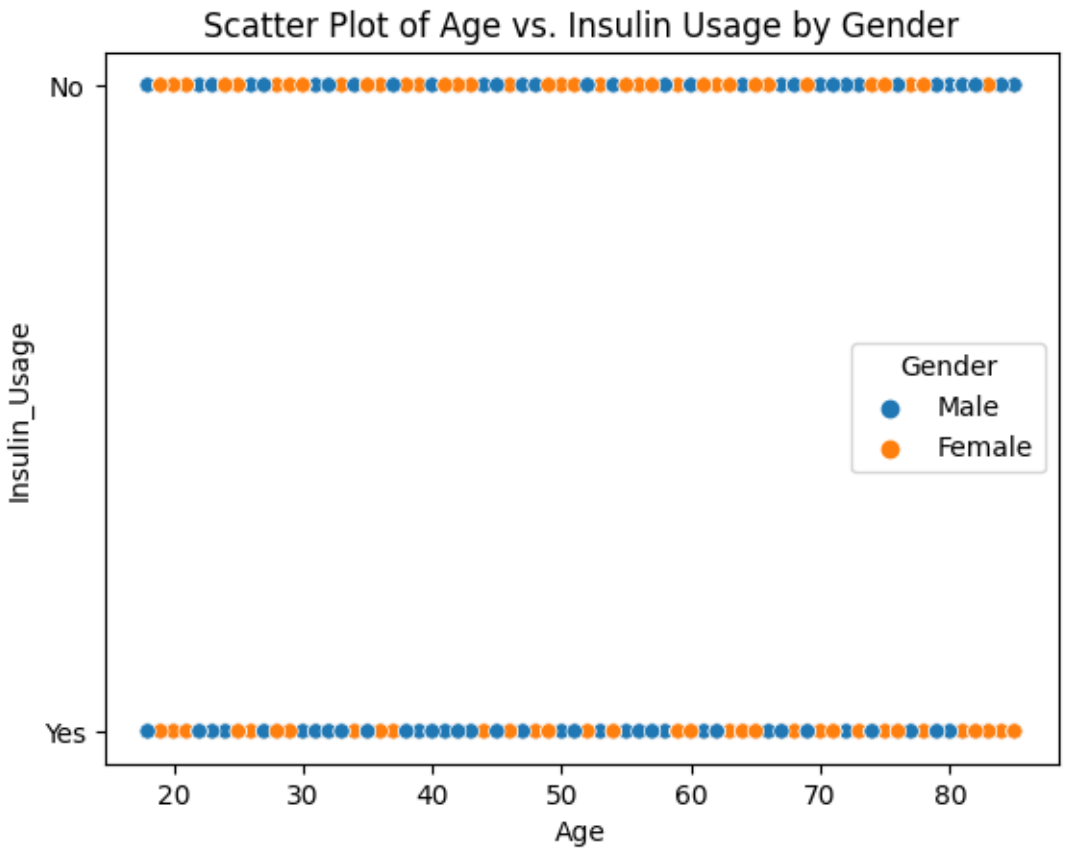


Q2. How does the scatter plot of age versus insulin usage, differentiated by gender, provide insights into the distribution of insulin usage across different age groups?

The scatter plot provides a visual exploration of age-related patterns in insulin usage, and the inclusion of gender as a hue allows for a more comprehensive analysis of how these patterns may vary between different demographic groups.

```
sns.scatterplot(x='Age', y='Insulin_Usage', hue='Gender', data=df)
plt.title('Scatter Plot of Age vs. Insulin Usage by Gender')
```

```
Text(0.5, 1.0, 'Scatter Plot of Age vs. Insulin Usage by Gender')
```




```
Diet_Type_distribution = df['Diet_Type'].value_counts()
print(Diet_Type_distribution)
```

```
High Protein    675
Balanced        664
High Carb       661
Name: Diet_Type, dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Patient_ID         2000 non-null   int64
 1   Age                2000 non-null   int64
 2   Gender             2000 non-null   object
 3   BMI                2000 non-null   float64
 4   Blood_Pressure     2000 non-null   object
 5   Glucose_Level      1789 non-null   float64
```

```
 6   Cholesterol_Level  1917 non-null   float64
 7   Physical_Activity  2000 non-null   float64
 8   Diet_Type          2000 non-null   object
 9   Insulin_Usage      2000 non-null   object
 10  HbA1c_Level        1770 non-null   float64
dtypes: float64(5), int64(2), object(4)
memory usage: 172.0+ KB
```

```
DeprecationWarning: `set_matplotlib_formats` is deprecated since IPython 7.23,
directly use `matplotlib_inline.backend_inline.set_matplotlib_formats()`
  set_matplotlib_formats('pdf', 'svg')
```