diabetes-patients

January 6, 2024

DIABETES PATIENTS

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import math
```

Import Dataset

[ ]:
```
data = pd.read_csv(r"/diabetes.csv")
```

[ ]:
```
#Copy the dataset
df = data.copy()
```

Data Exploration

[ ]:
```
df.head()
```

[ ]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

[ ]:
```
df.tail
```

[ ]:
```
<bound method NDFrame.tail of      Pregnancies  Glucose  BloodPressure
SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
```

```
1              1       85             66             29        0  26.6
2              8      183             64              0        0  23.3
3              1       89             66             23       94  28.1
4              0      137             40             35      168  43.1
..           ...      ...            ...            ...      ...   ...
763           10      101             76             48      180  32.9
764            2      122             70             27        0  36.8
765            5      121             72             23      112  26.2
766            1      126             60              0        0  30.1
767            1       93             70             31        0  30.4

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
1                       0.351   31        0
2                       0.672   32        1
3                       0.167   21        0
4                       2.288   33        1
..                        ...  ...      ...
763                     0.171   63        0
764                     0.340   27        0
765                     0.245   30        0
766                     0.349   47        1
767                     0.315   23        0

[768 rows x 9 columns]>
```

Number of Rows and Columns in Dataset

[ ]:
```
df.shape
print("Total Number of Rows in Dataset :",data.shape[0])
print("Total Number of Columns in Dataset:",data.shape[1])
#Total Number of Rows in Dataset : 768
#Total Number of Columns in Dataset: 9
df.info()
```

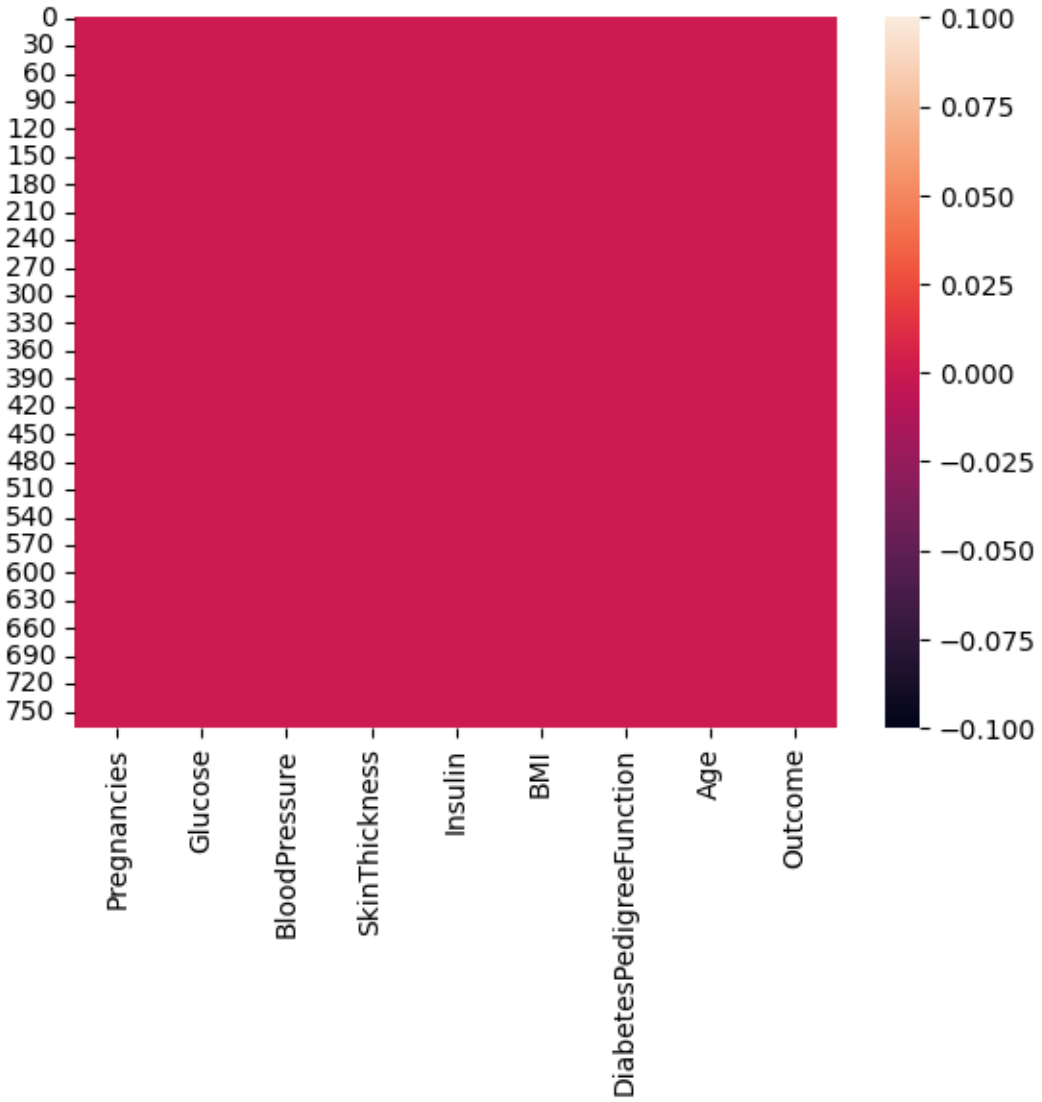
```
Total Number of Rows in Dataset : 768
Total Number of Columns in Dataset: 9
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
```

```
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Check Missing Values in Dataset

[ ]:
```
sns.heatmap(data.isnull())
```

[ ]:
```
<Axes: >
```



Overall Statistics About The Dataset

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

[ ]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin
count	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479
std	3.369578	31.972618	19.355807	15.952218	115.244002
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000
75%	6.000000	140.250000	80.000000	32.000000	127.250000
max	17.000000	199.000000	122.000000	99.000000	846.000000

	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000
mean	31.992578	0.471876	33.240885	0.348958
std	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.078000	21.000000	0.000000
25%	27.300000	0.243750	24.000000	0.000000
50%	32.000000	0.372500	29.000000	0.000000
75%	36.600000	0.626250	41.000000	1.000000
max	67.100000	2.420000	81.000000	1.000000

```
<google.colab._quickchart_helpers.SectionTitle at 0x7f585836a710>
```

```
from matplotlib import pyplot as plt
_df_0['Pregnancies'].plot(kind='hist', bins=20, title='Pregnancies')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_1['Glucose'].plot(kind='hist', bins=20, title='Glucose')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_2['BloodPressure'].plot(kind='hist', bins=20, title='BloodPressure')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_3['SkinThickness'].plot(kind='hist', bins=20, title='SkinThickness')
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7f5855f51f00>
```

```
from matplotlib import pyplot as plt
_df_4.plot(kind='scatter', x='Pregnancies', y='Glucose', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_5.plot(kind='scatter', x='Glucose', y='BloodPressure', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
```

```
_df_6.plot(kind='scatter', x='BloodPressure', y='SkinThickness', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_7.plot(kind='scatter', x='SkinThickness', y='Insulin', s=32, alpha=.8)
plt.gca().spines[['top', 'right',]].set_visible(False)
```

```
<google.colab._quickchart_helpers.SectionTitle at 0x7f58583691e0>
```

```
from matplotlib import pyplot as plt
_df_8['Pregnancies'].plot(kind='line', figsize=(8, 4), title='Pregnancies')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_9['Glucose'].plot(kind='line', figsize=(8, 4), title='Glucose')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_10['BloodPressure'].plot(kind='line', figsize=(8, 4), title='BloodPressure')
plt.gca().spines[['top', 'right']].set_visible(False)
```

```
from matplotlib import pyplot as plt
_df_11['SkinThickness'].plot(kind='line', figsize=(8, 4), title='SkinThickness')
plt.gca().spines[['top', 'right']].set_visible(False)
```

CO RELATION MATRIX

[ ]:
```
corelation = data.corr()
print(corelation)
```

```
                          Pregnancies   Glucose  BloodPressure  SkinThickness  \
Pregnancies                  1.000000  0.129459       0.141282      -0.081672
Glucose                      0.129459  1.000000       0.152590       0.057328
BloodPressure                0.141282  0.152590       1.000000       0.207371
SkinThickness               -0.081672  0.057328       0.207371       1.000000
Insulin                     -0.073535  0.331357       0.088933       0.436783
BMI                          0.017683  0.221071       0.281805       0.392573
DiabetesPedigreeFunction    -0.033523  0.137337       0.041265       0.183928
Age                          0.544341  0.263514       0.239528      -0.113970
Outcome                      0.221898  0.466581       0.065068       0.074752

                            Insulin       BMI  DiabetesPedigreeFunction  \
Pregnancies               -0.073535  0.017683                 -0.033523
Glucose                    0.331357  0.221071                  0.137337
BloodPressure              0.088933  0.281805                  0.041265
SkinThickness              0.436783  0.392573                  0.183928
Insulin                    1.000000  0.197859                  0.185071
BMI                        0.197859  1.000000                  0.140647
DiabetesPedigreeFunction   0.185071  0.140647                  1.000000
Age                       -0.042163  0.036242                  0.033561
Outcome                    0.130548  0.292695                  0.173844
```

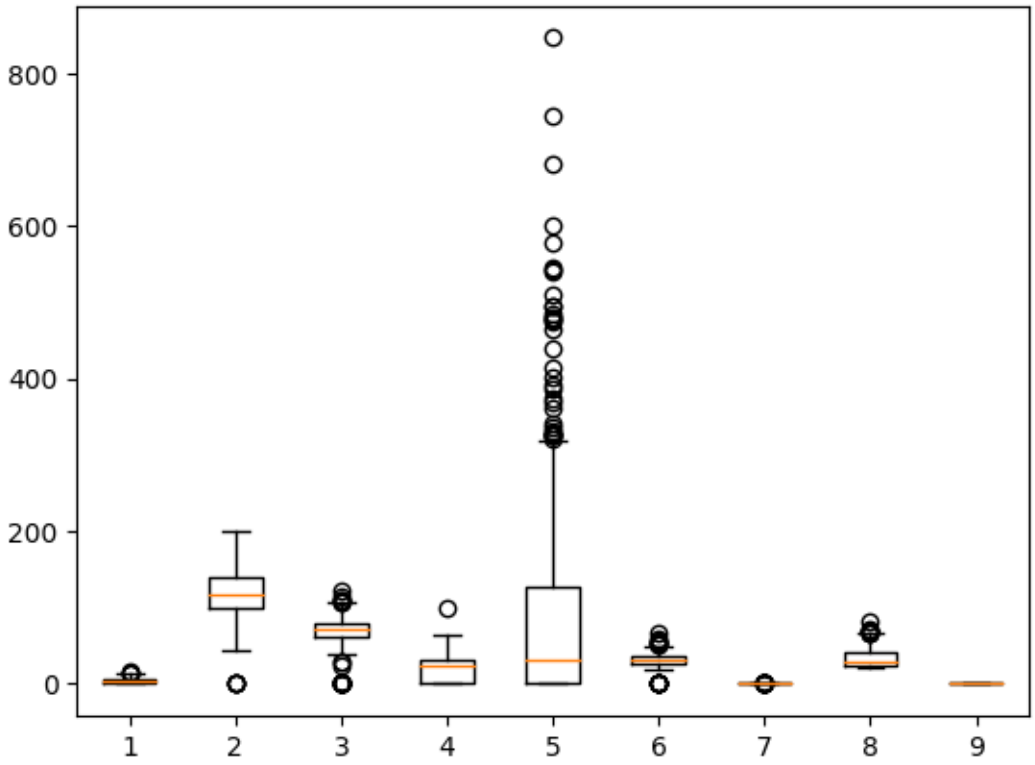
```
                                Age   Outcome
Pregnancies                0.544341  0.221898
Glucose                    0.263514  0.466581
BloodPressure              0.239528  0.065068
SkinThickness             -0.113970  0.074752
Insulin                   -0.042163  0.130548
BMI                        0.036242  0.292695
DiabetesPedigreeFunction   0.033561  0.173844
Age                        1.000000  0.238356
Outcome                    0.238356  1.000000
```

Checking Outliers

[18]:
```
plt.boxplot(data)
```

[18]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x7f58558a60e0>,
  <matplotlib.lines.Line2D at 0x7f58558a6bf0>,
  <matplotlib.lines.Line2D at 0x7f58558a5240>,
  <matplotlib.lines.Line2D at 0x7f58558c7f70>,
  <matplotlib.lines.Line2D at 0x7f58558c4220>,
  <matplotlib.lines.Line2D at 0x7f58558c4eb0>,
  <matplotlib.lines.Line2D at 0x7f58558c76d0>,
  <matplotlib.lines.Line2D at 0x7f58558c4100>,
  <matplotlib.lines.Line2D at 0x7f58558c46d0>,
  <matplotlib.lines.Line2D at 0x7f58558c4f10>,
  <matplotlib.lines.Line2D at 0x7f58545fd060>,
  <matplotlib.lines.Line2D at 0x7f58545fe6e0>,
  <matplotlib.lines.Line2D at 0x7f58545fd3f0>,
  <matplotlib.lines.Line2D at 0x7f58545fc790>,
  <matplotlib.lines.Line2D at 0x7f58545fdf60>,
  <matplotlib.lines.Line2D at 0x7f58545fff70>,
  <matplotlib.lines.Line2D at 0x7f585581c4c0>,
  <matplotlib.lines.Line2D at 0x7f585581ca00>],
 'caps': [<matplotlib.lines.Line2D at 0x7f58558a6620>,
  <matplotlib.lines.Line2D at 0x7f58558a5de0>,
  <matplotlib.lines.Line2D at 0x7f58558c6b00>,
  <matplotlib.lines.Line2D at 0x7f58558c6d10>,
  <matplotlib.lines.Line2D at 0x7f58558c5c90>,
  <matplotlib.lines.Line2D at 0x7f58558c6aa0>,
  <matplotlib.lines.Line2D at 0x7f58558c6a10>,
  <matplotlib.lines.Line2D at 0x7f58558c4850>,
  <matplotlib.lines.Line2D at 0x7f58545fc280>,
  <matplotlib.lines.Line2D at 0x7f58545ff6a0>,
  <matplotlib.lines.Line2D at 0x7f58545fe5c0>,
  <matplotlib.lines.Line2D at 0x7f58545fc460>,
  <matplotlib.lines.Line2D at 0x7f58545fc3a0>,
  <matplotlib.lines.Line2D at 0x7f58545ff2b0>,
```

```
 <matplotlib.lines.Line2D at 0x7f58545feb00>,
 <matplotlib.lines.Line2D at 0x7f58545fd720>,
 <matplotlib.lines.Line2D at 0x7f585581cc40>,
 <matplotlib.lines.Line2D at 0x7f585581d4b0>],
'boxes': [<matplotlib.lines.Line2D at 0x7f58558a7520>,
 <matplotlib.lines.Line2D at 0x7f58558a7eb0>,
 <matplotlib.lines.Line2D at 0x7f58558c6b90>,
 <matplotlib.lines.Line2D at 0x7f58558c66e0>,
 <matplotlib.lines.Line2D at 0x7f58558c6170>,
 <matplotlib.lines.Line2D at 0x7f58545fd4b0>,
 <matplotlib.lines.Line2D at 0x7f58545ff8e0>,
 <matplotlib.lines.Line2D at 0x7f58545fe5f0>,
 <matplotlib.lines.Line2D at 0x7f585581ecb0>],
'medians': [<matplotlib.lines.Line2D at 0x7f58558a43a0>,
 <matplotlib.lines.Line2D at 0x7f58558c7d00>,
 <matplotlib.lines.Line2D at 0x7f58558c4f40>,
 <matplotlib.lines.Line2D at 0x7f58558c50f0>,
 <matplotlib.lines.Line2D at 0x7f58545fc850>,
 <matplotlib.lines.Line2D at 0x7f58545fcf10>,
 <matplotlib.lines.Line2D at 0x7f58545feef0>,
 <matplotlib.lines.Line2D at 0x7f58545fd360>,
 <matplotlib.lines.Line2D at 0x7f585581db10>],
'fliers': [<matplotlib.lines.Line2D at 0x7f58558a5f00>,
 <matplotlib.lines.Line2D at 0x7f58558c6c50>,
 <matplotlib.lines.Line2D at 0x7f58558c7eb0>,
 <matplotlib.lines.Line2D at 0x7f58558c5090>,
 <matplotlib.lines.Line2D at 0x7f58545fe2f0>,
 <matplotlib.lines.Line2D at 0x7f58545ffa90>,
 <matplotlib.lines.Line2D at 0x7f58545ff1f0>,
 <matplotlib.lines.Line2D at 0x7f585581f580>,
 <matplotlib.lines.Line2D at 0x7f585581d8a0>],
'means': []}
```



Managing Outliers in Dataset

[ ]:
```
#Create a function to handle Outliers
def remove_outliers(data, column_name):
Q1 = data[column_name].quantile(0.25)
Q3 = data[column_name].quantile(0.75)
IQR = Q3 - Q1
upper_limit = Q3 + 1.5 * IQR
lower_limit = Q1 - 1.5 * IQR
data[column_name] = data[column_name].clip(lower=lower_limit, upper=upper_limit)
return data
#Handle outliers using "remove_outliers" function
df = remove_outliers(df, 'Pregnancies')
df = remove_outliers(df, 'Glucose')
df = remove_outliers(df, 'BloodPressure')
df = remove_outliers(df, 'SkinThickness')
df = remove_outliers(df, 'Insulin')
df = remove_outliers(df, 'BMI')
import plotly.graph_objs as go
# Create a list to store the box plot traces
box_traces = []
for column in df.columns:
```

```
if column != 'Outcome': # Exclude 'Outcome' if it's the target variable
trace = go.Box(y=df[column], name=column)
box_traces.append(trace)
# Create a layout
layout = go.Layout(title='Box Plots for Dataset Columns')
# Create a figure and add the traces and layout
fig = go.Figure(data=box_traces, layout=layout)
# Show the figure
fig.show()
```

After handling outliers , the datatype of some columns has changed to float. We also need to convert them back to int32.

[21]:
```
df['Pregnancies']=round(df['Pregnancies'].astype('int32'))
df['Glucose']=round(df['Glucose'].astype('int32'))
df['Insulin']=round(df['Insulin'].astype('int32'))
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int32
 1   Glucose                   768 non-null    int32
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int32
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int32(3), int64(4)
memory usage: 45.1 KB
```

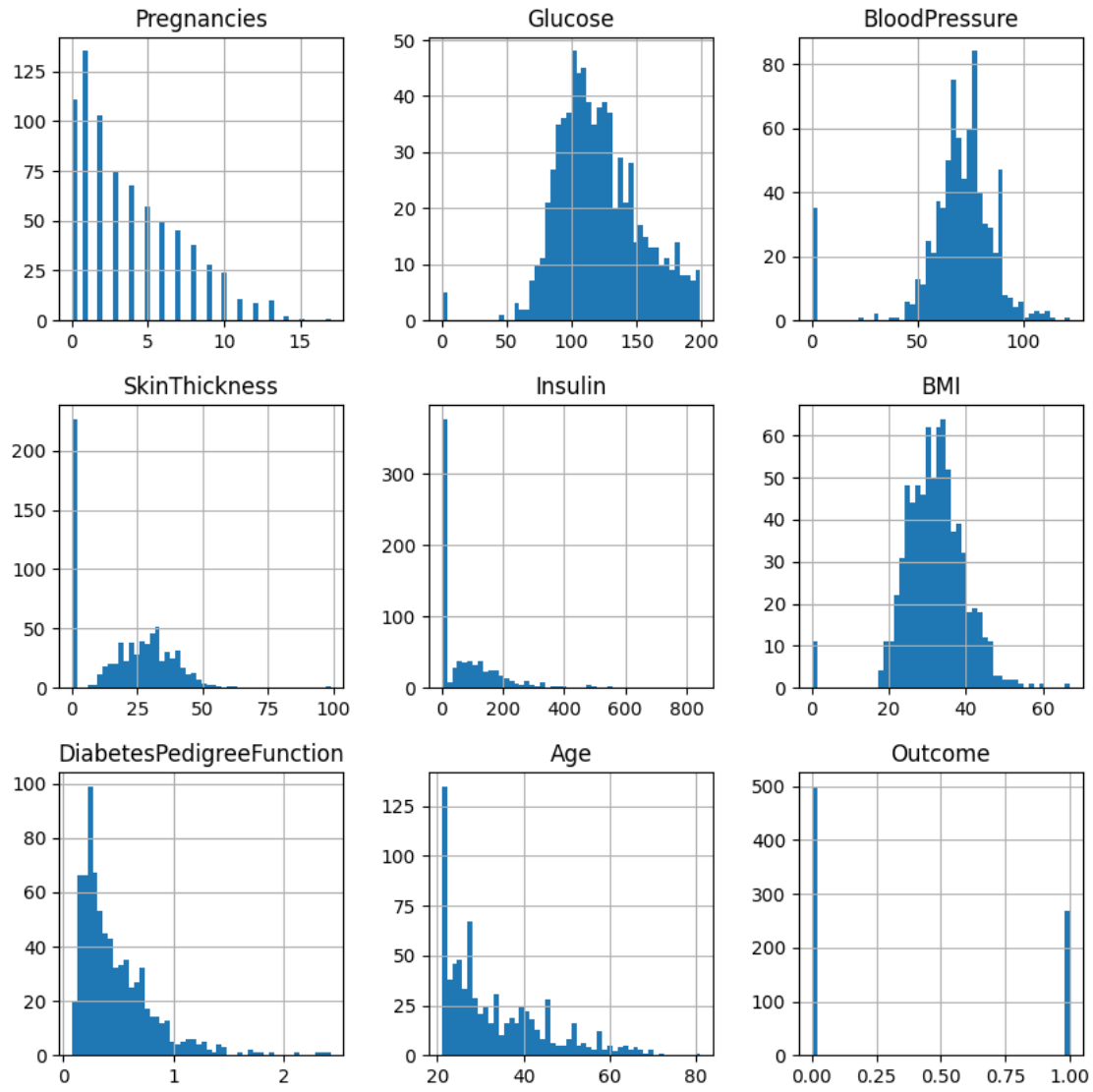
[26]:
```
df.columns
```

[26]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'],
      dtype='object')
```

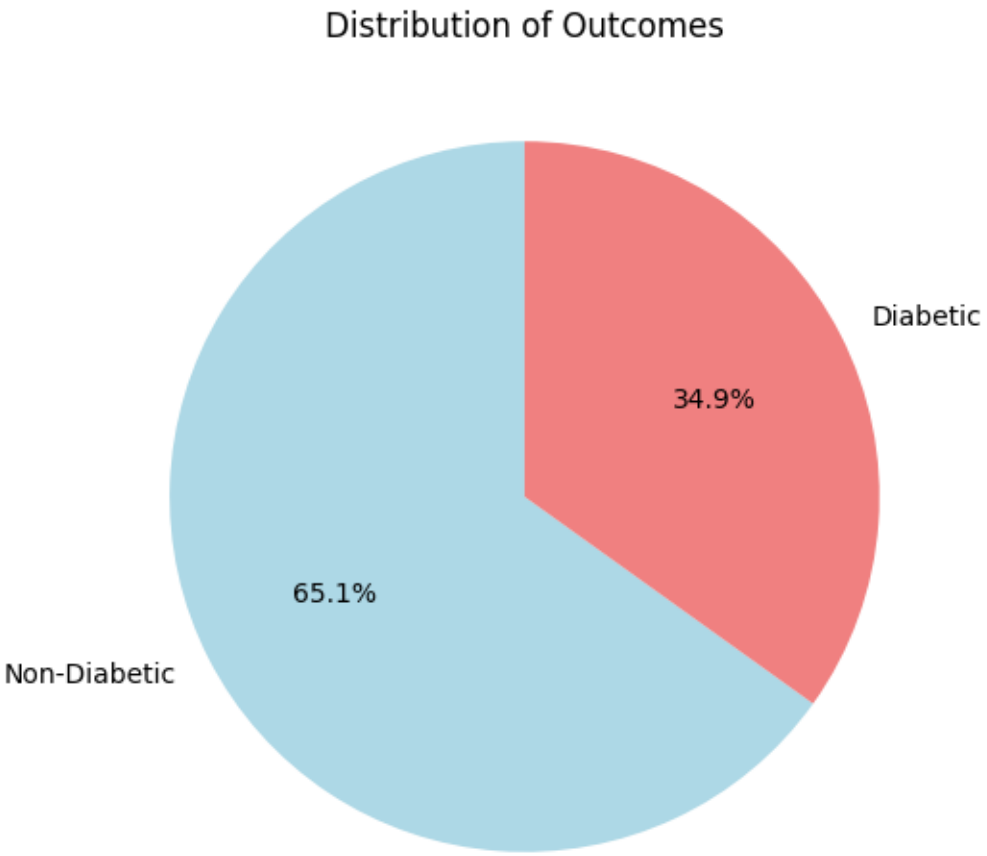
Visualizing the Dataset for Better Understanding

[25]:
```
import pandas as pd
import matplotlib.pyplot as plt
# Plot the histogram
fig, ax = plt.subplots(figsize=(10, 10))
df.hist(bins=50, ax=ax)
```

```
plt.show()
```



[27]:
```
# Count the occurrences of each outcome value
outcome_counts = df['Outcome'].value_counts()
# Create a pie chart
plt.figure(figsize=(6, 6))
plt.pie(outcome_counts, labels=['Non-Diabetic', 'Diabetic'], autopct='%1.1f%%',␣
  ↪startangle=90, colors=['lightblue', 'lightcoral'])
plt.title('Distribution of Outcomes')
plt.show()
```

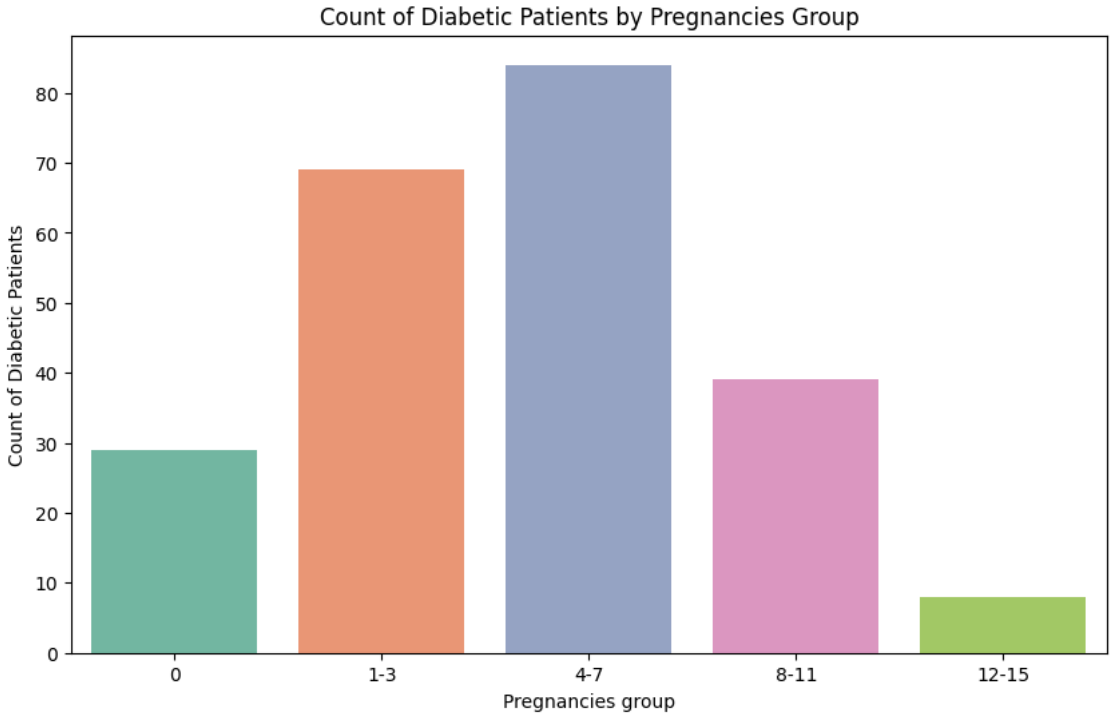


Converting Numerical Features into Categorical Features for Data Clarity:

Create bins for the 'Pregnancies' column

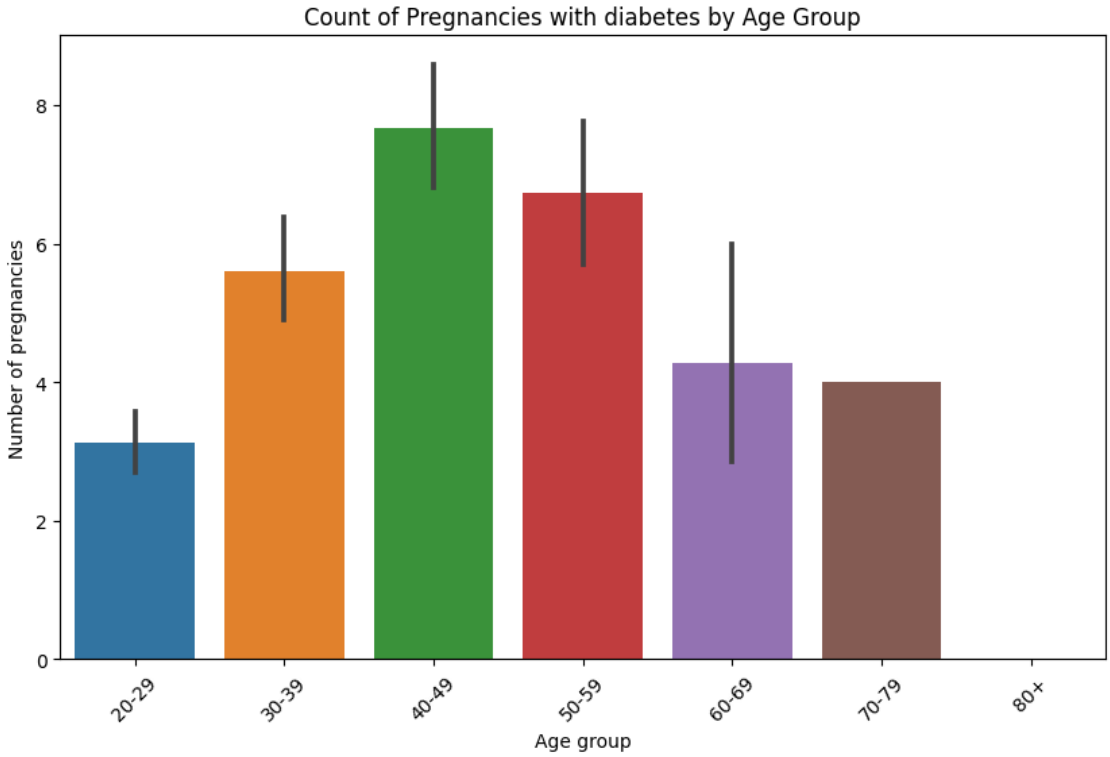
[28]:
```
pregnancies_bins = [0, 1, 4, 8, 12, 16]
pregnancies_labels = ['0', '1-3', '4-7', '8-11', '12-15']
df['PregnanciesGroup'] = pd.cut(df['Pregnancies'], bins=pregnancies_bins,␣
 ↪labels=pregnancies_labels)
# Filter the dataset to include only records with 'Outcome' equal to 1␣
 ↪(Diabetic patients)
diabetic_df = df[df['Outcome'] == 1]
# Create a bar chart for Diabetic patients with 'PregnanciesGroup' as the x-axis
plt.figure(figsize=(10, 6))
sns.countplot(data=diabetic_df, x='PregnanciesGroup', order=pregnancies_labels,␣
 ↪palette="Set2")
plt.xlabel('Pregnancies group')
plt.ylabel('Count of Diabetic Patients')
plt.title('Count of Diabetic Patients by Pregnancies Group')
```

```
plt.show()
```

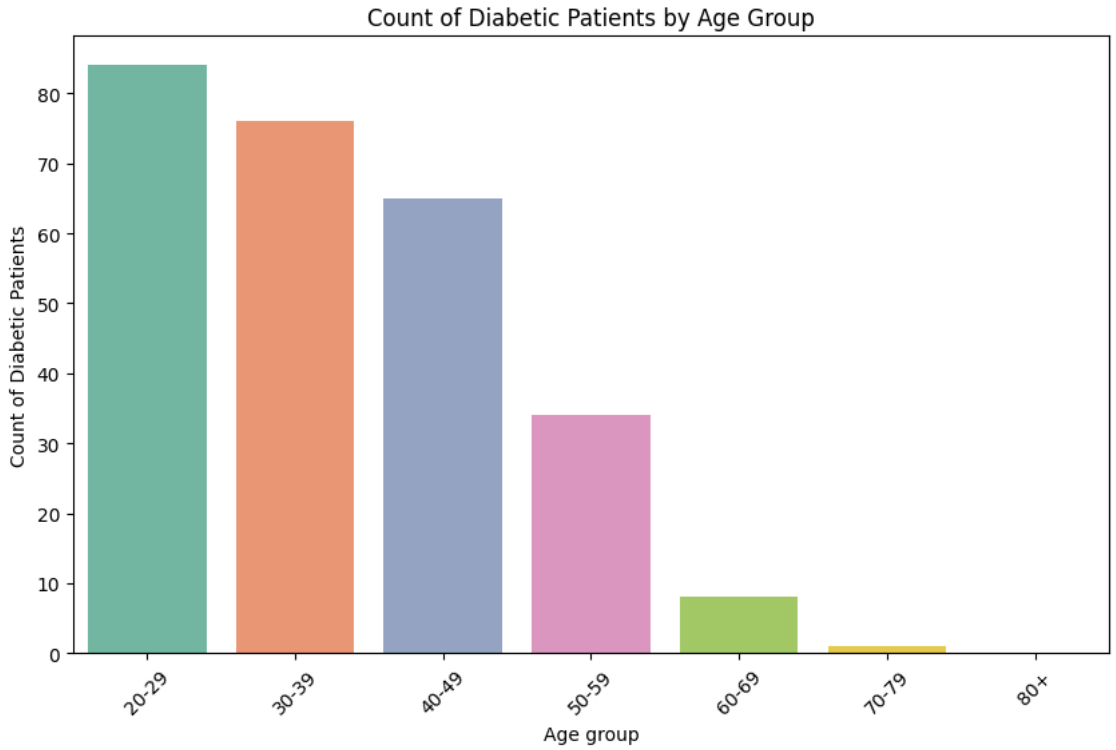


Create age groups based on the 'Age' column

[31]:
```
new_df = df[(df['Outcome'] == 1) & (df['Pregnancies'] > 0)]
# Create a bar chart with 'Outcome' as hue
plt.figure(figsize=(10, 6))
ax = sns.barplot(data=new_df, x='AgeGroup', y='Pregnancies')
plt.xlabel('Age group')
plt.ylabel('Number of pregnancies')
plt.title('Count of Pregnancies with diabetes by Age Group')
plt.xticks(rotation=45)
plt.show()
```



[30]:
```
bins = [20, 30, 40, 50, 60, 70, 80, 200]
labels = ['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)
# Filter the dataset to include only records with 'Outcome' equal to 1␣
 ↪(Diabetic patients)
diabetic_df = df[df['Outcome'] == 1]
# Create a bar chart for Diabetic patients with age groups
plt.figure(figsize=(10, 6))
sns.countplot(data=diabetic_df, x='AgeGroup', order=labels, palette="Set2")
plt.xlabel('Age group')
plt.ylabel('Count of Diabetic Patients')
plt.title('Count of Diabetic Patients by Age Group')
plt.xticks(rotation=45)
plt.show()
```



Define the bins and labels for 'BloodPressure'

[32]:
```
# Define the bins and labels for 'BloodPressure'
blood_pressure_bins = [0, 80, 89, 99, 119, 1000] # Adjust the boundaries as␣
 ↪needed
blood_pressure_labels = ['Low', 'Normal', 'Prehypertension', 'Stage 1␣
 ↪hypertension', 'Stage 2 hypertension']
# Create a new column 'BloodPressureCategory' based on the bins and labels
df['BloodPressureCategory'] = pd.cut(df['BloodPressure'],␣
 ↪bins=blood_pressure_bins, labels=blood_pressure_labels,right=False)
df.head()
```

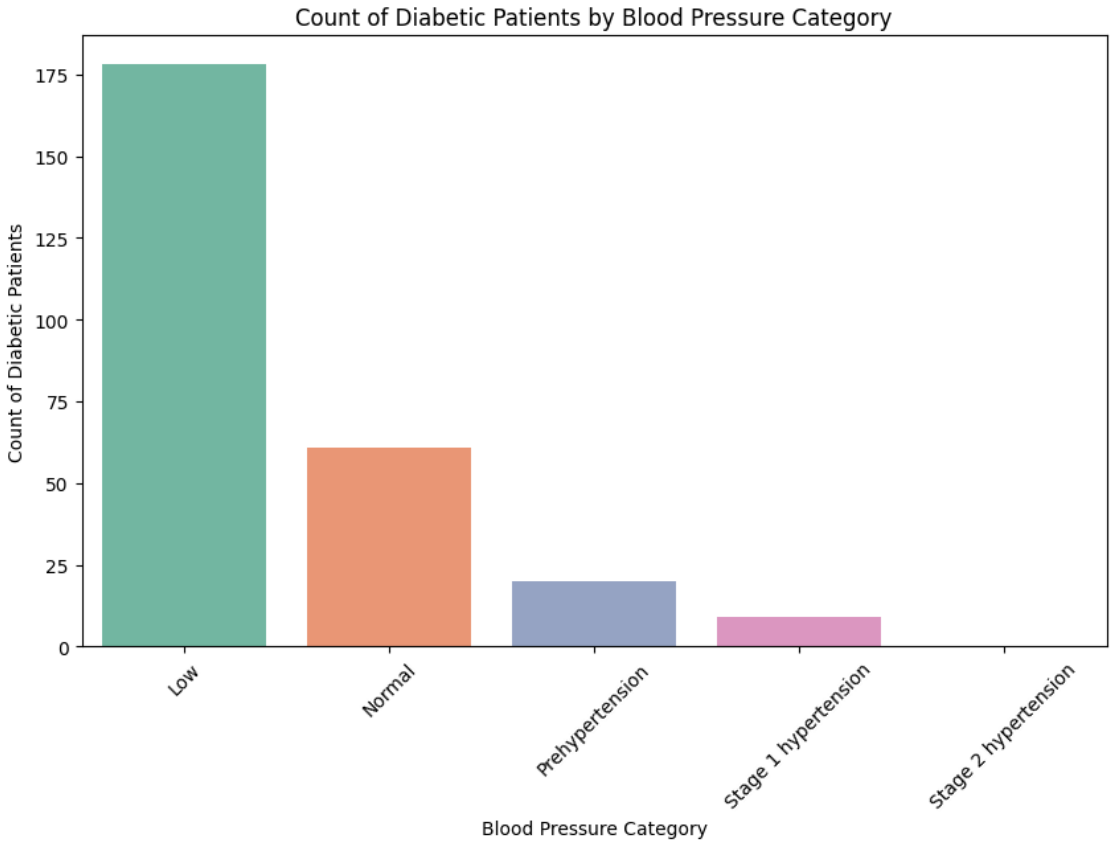
[32]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	
2	8	183	64	0	0	23.3	
3	1	89	66	23	94	28.1	
4	0	137	40	35	168	43.1	

	DiabetesPedigreeFunction	Age	Outcome	PregnanciesGroup	AgeGroup	\
0	0.627	50	1	4-7	50-59	
1	0.351	31	0	0	30-39	
2	0.672	32	1	4-7	30-39	

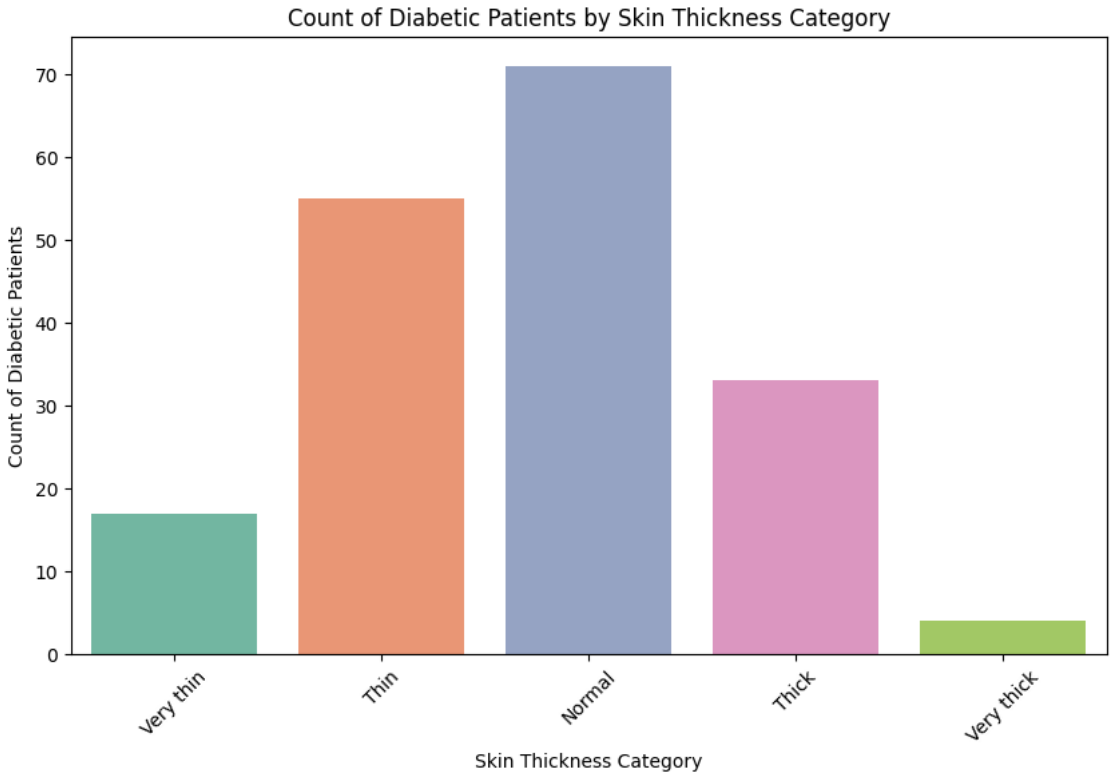
```
3                     0.167   21          0                 0     20-29
4                     2.288   33          1               NaN     30-39

  BloodPressureCategory
0                   Low
1                   Low
2                   Low
3                   Low
4                   Low
```

[33]:
```
# Filter the dataset to include only records with 'Outcome' equal to 1␣
 ↪(Diabetic patients)
diabetic_df = df[df['Outcome'] == 1]
# Create a bar chart for Diabetic patients with 'BloodPressureCategory' as the␣
 ↪x-axis
plt.figure(figsize=(10, 6))
sns.countplot(data=diabetic_df, x='BloodPressureCategory',␣
 ↪order=blood_pressure_labels, palette="Set2")
plt.xlabel('Blood Pressure Category')
plt.ylabel('Count of Diabetic Patients')
plt.title('Count of Diabetic Patients by Blood Pressure Category')
plt.xticks(rotation=45)
plt.show()
```



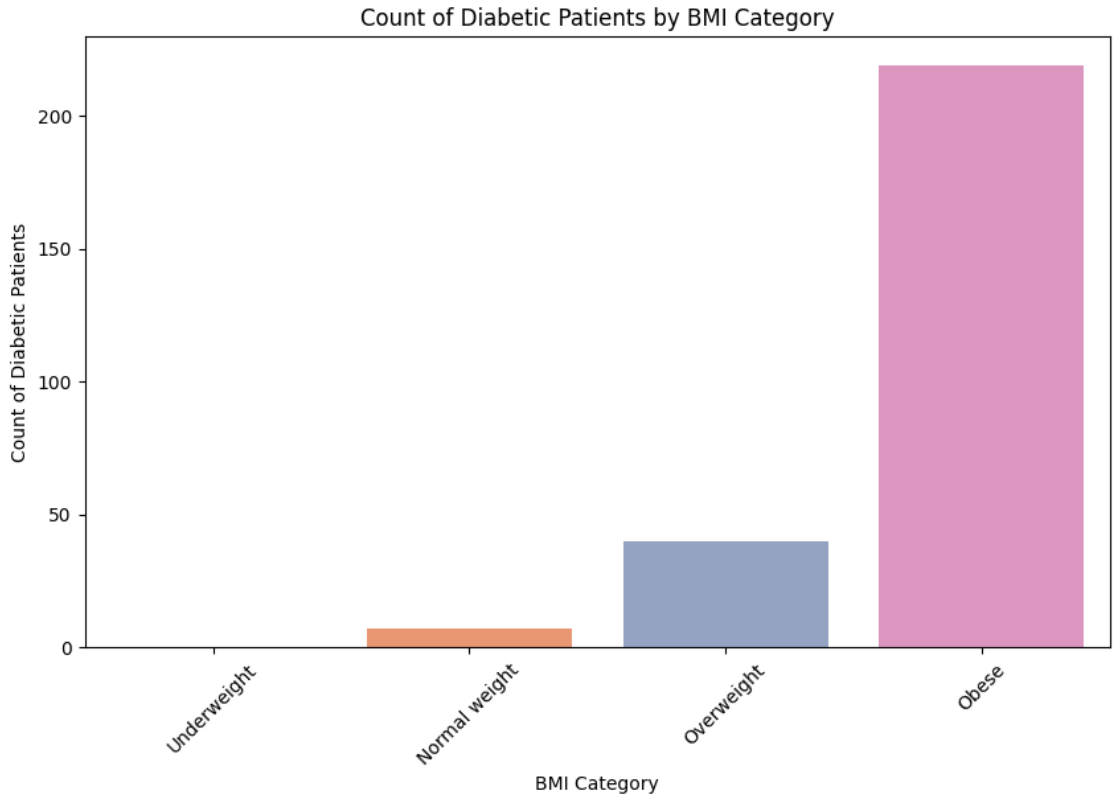
Define the bins and labels for 'SkinThickness'

[34]:
```
skin_thickness_bins = [0, 20, 30, 40, 50, 100]
skin_thickness_labels = ['Very thin', 'Thin', 'Normal', 'Thick', 'Very thick']
# Create a new column 'SkinThicknessCategory' based on the bins and labels
df['SkinThicknessCategory'] = pd.cut(df['SkinThickness'],␣
 ↪bins=skin_thickness_bins, labels=skin_thickness_labels)
# Filter the dataset to include only records with 'Outcome' equal to 1␣
 ↪(Diabetic patients)
diabetic_df = df[df['Outcome'] == 1]
# Create a bar chart for Diabetic patients with 'SkinThicknessCategory' as the␣
 ↪x-axis
plt.figure(figsize=(10, 6))
sns.countplot(data=diabetic_df, x='SkinThicknessCategory',␣
 ↪order=skin_thickness_labels, palette="Set2")
plt.xlabel('Skin Thickness Category')
plt.ylabel('Count of Diabetic Patients')
plt.title('Count of Diabetic Patients by Skin Thickness Category')
plt.xticks(rotation=45)
plt.show()
```



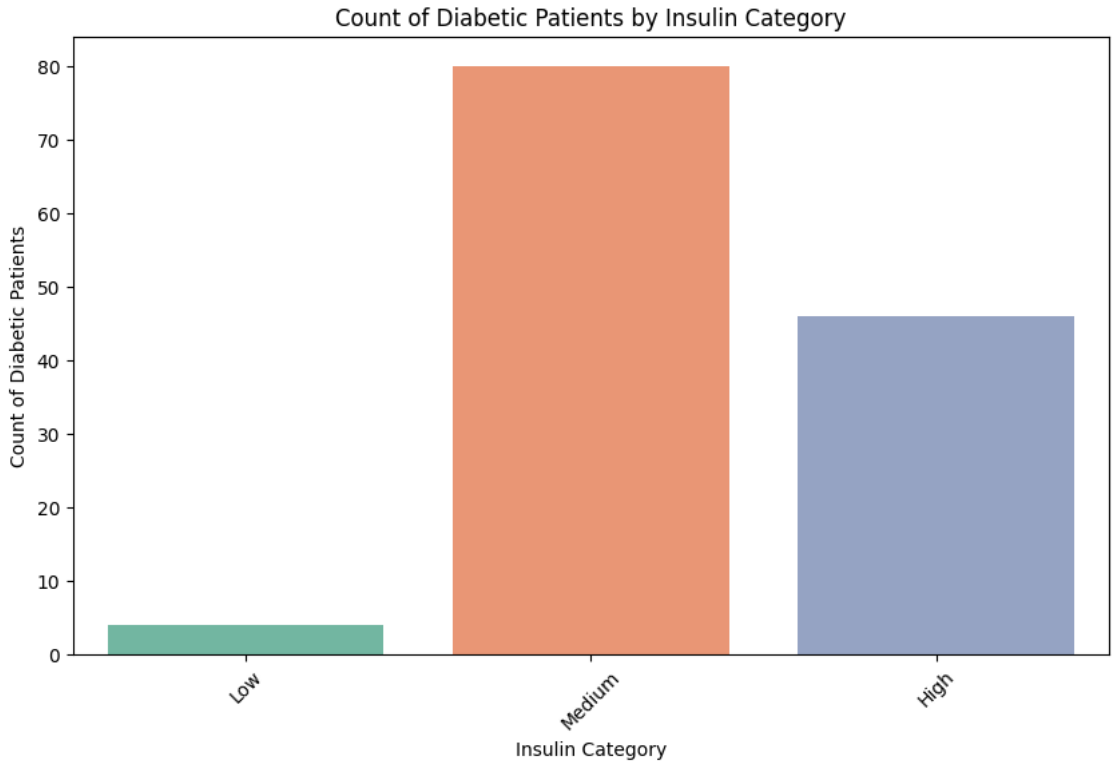
Define the custom bins and labels for 'BMI'

[35]:
```
# Define the custom bins and labels for 'BMI'
bmi_bins = [0, 18.5, 24.9, 29.9, 1000]
bmi_labels = ['Underweight', 'Normal weight', 'Overweight', 'Obese']
# Create a new column 'BMICategory' based on the custom bins and labels
df['BMICategory'] = pd.cut(df['BMI'], bins=bmi_bins, labels=bmi_labels)
# Filter the dataset to include only records with 'Outcome' equal to 1␣
 ↪(Diabetic patients)
diabetic_df = df[df['Outcome'] == 1]
# Create a bar chart for Diabetic patients with 'BMICategory' as the x-axis
plt.figure(figsize=(10, 6))
sns.countplot(data=diabetic_df, x='BMICategory', order=bmi_labels,␣
 ↪palette="Set2")
plt.xlabel('BMI Category')
plt.ylabel('Count of Diabetic Patients')
plt.title('Count of Diabetic Patients by BMI Category')
plt.xticks(rotation=45)
plt.show()
```



Define the custom bins and labels for 'Insulin

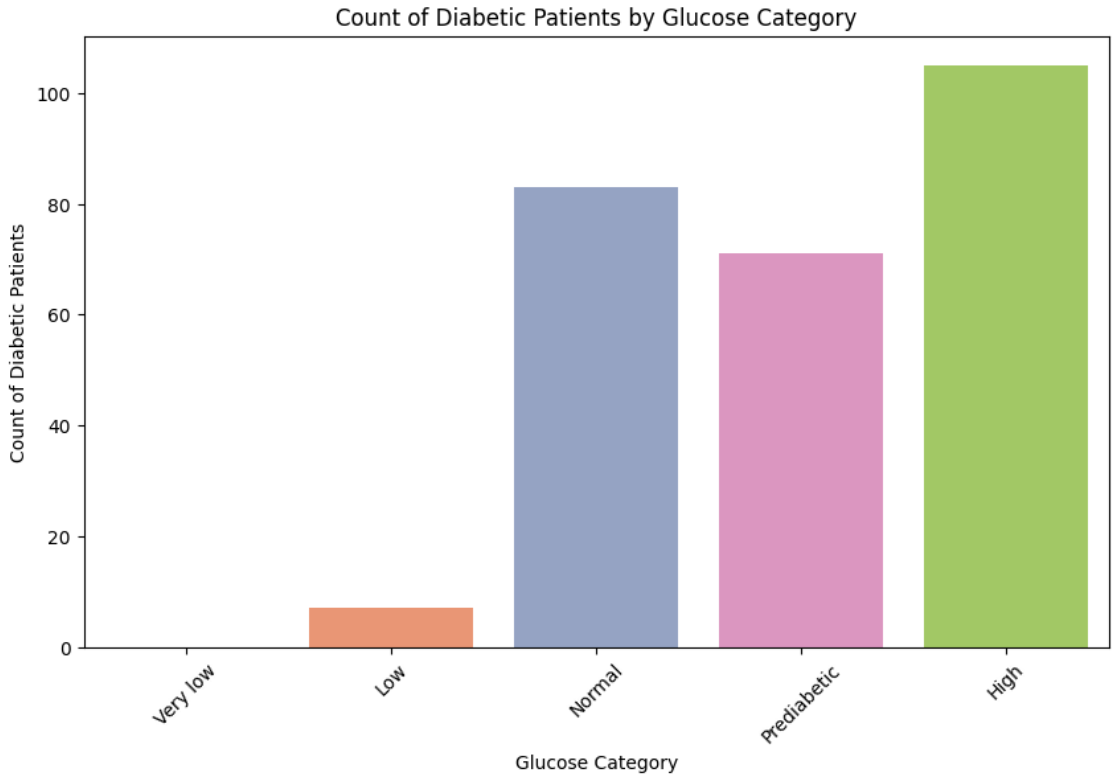
[37]:
```
insulin_bins = [0, 50, 200, 10000]
insulin_labels = ['Low', 'Medium', 'High']
# Create a new column 'InsulinCategory' based on the custom bins and labels
df['InsulinCategory'] = pd.cut(df['Insulin'], bins=insulin_bins,␣
 ↪labels=insulin_labels)
# Filter the dataset to include only records with 'Outcome' equal to 1␣
 ↪(Diabetic patients)
diabetic_df = df[df['Outcome'] == 1]
# Create a bar chart for Diabetic patients with 'InsulinCategory' as the x-axis
plt.figure(figsize=(10, 6))
sns.countplot(data=diabetic_df, x='InsulinCategory', order=insulin_labels,␣
 ↪palette="Set2")
plt.xlabel('Insulin Category')
plt.ylabel('Count of Diabetic Patients')
plt.title('Count of Diabetic Patients by Insulin Category')
plt.xticks(rotation=45)
plt.show()
```



[ ]:

Define the custom bins and labels for 'Glucose'

[38]:
```
glucose_bins = [0, 75, 90, 125, 150, 1000] # Adjust the boundaries as needed
glucose_labels = ['Very low', 'Low', 'Normal', 'Prediabetic', 'High']
# Create a new column 'GlucoseCategory' based on the custom bins and labels
df['GlucoseCategory'] = pd.cut(df['Glucose'], bins=glucose_bins,␣
 ↪labels=glucose_labels)
# Filter the dataset to include only records with 'Outcome' equal to 1␣
 ↪(Diabetic patients)
diabetic_df = df[df['Outcome'] == 1]
# Create a bar chart for Diabetic patients with 'GlucoseCategory' as the x-axis
plt.figure(figsize=(10, 6))
sns.countplot(data=diabetic_df, x='GlucoseCategory', order=glucose_labels,␣
 ↪palette="Set2")
plt.xlabel('Glucose Category')
plt.ylabel('Count of Diabetic Patients')
plt.title('Count of Diabetic Patients by Glucose Category')
plt.xticks(rotation=45)
plt.show()
```



Data Wrangling:

Seprate Independent Variable(X) and Dependent Variable(y)

[40]:
```
df.columns
```

[40]:
```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',
       'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome', 'PregnanciesGroup',
       'AgeGroup', 'BloodPressureCategory', 'SkinThicknessCategory',
       'BMICategory', 'InsulinCategory', 'GlucoseCategory'],
      dtype='object')
```

[41]:
```
#Independent Variables
X = df[['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin',␣
 ↪'BMI', 'DiabetesPedigreeFunction', 'Age']]
#Target variable
y = df['Outcome']
```

[42]:
```
df.head()
```

[42]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	\
0	6	148	72	35	0	33.6	
1	1	85	66	29	0	26.6	

```
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome PregnanciesGroup AgeGroup  \
0                     0.627   50        1              4-7    50-59
1                     0.351   31        0                0    30-39
2                     0.672   32        1              4-7    30-39
3                     0.167   21        0                0    20-29
4                     2.288   33        1              NaN    30-39

  BloodPressureCategory SkinThicknessCategory    BMICategory InsulinCategory  \
0                   Low                Normal          Obese             NaN
1                   Low                  Thin     Overweight             NaN
2                   Low                   NaN  Normal weight             NaN
3                   Low                  Thin     Overweight          Medium
4                   Low                Normal          Obese          Medium

  GlucoseCategory
0     Prediabetic
1             Low
2            High
3             Low
4     Prediabetic
```

[45]:
```
X.head()
```

[45]:
```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age
0                     0.627   50
1                     0.351   31
2                     0.672   32
3                     0.167   21
4                     2.288   33
```

[44]:
```
y.head()
```

[44]:
```
0    1
1    0
2    1
3    0
```

```
4    1
Name: Outcome, dtype: int64
```

[46]:
```
#Split the data into training and testing sets.
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
```

Feature Scaling

[48]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train
```

[48]:
```
array([[-0.85811767,  0.06488386,  0.25332145, …, -0.51313743,
        -1.10316947, -0.27704152],
       [-0.85811767, -0.84697246,  0.66358026, …,  0.4081093 ,
        -0.71238555,  0.84376203],
       [-1.15412006, -0.87841578,  0.04819205, …,  1.49569224,
        -0.37742791, -1.05298243],
       …,
       [ 0.02988949,  0.09632718,  0.04819205, …, -0.3723914 ,
         1.96433735,  1.01619334],
       [-0.2661129 , -0.18666271,  0.25332145, …, -0.70506383,
        -1.08260189, -0.79433546],
       [ 0.02988949, -0.37532264, -0.15693736, …,  0.0114614 ,
        -0.01308802, -0.36325717]])
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: