Aim: Perform following data visualization and exploration on your selected dataset.

- Create bar graph, contingency table using any 2 features.
- Plot Scatter plot, box plot, Heatmap using seaborn.
- Create histogram and normalized Histogram.
- Describe what this graph and table indicates.
- Handle outlier using box plot and Inter quartile range

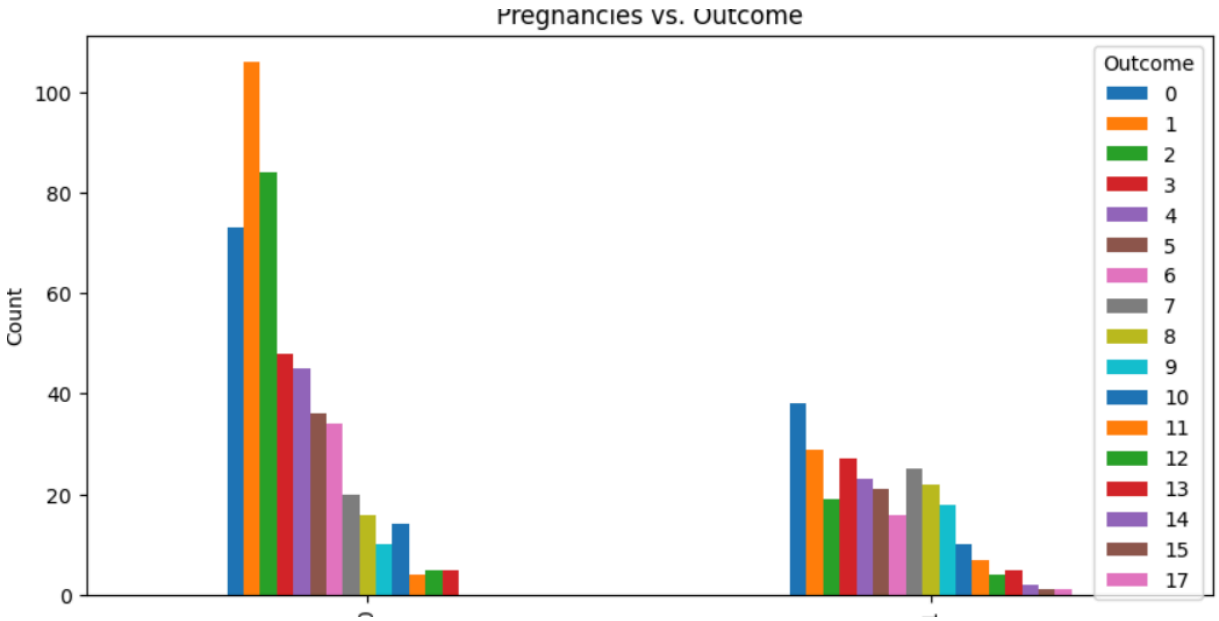
Steps:

1.**Create Bar Graph and Contingency Table**

Seaborn and Matplotlib are used together to create an effective bar graph. Matplotlib sets the figure size, labels, and titles to ensure clarity in the visualization. Seaborn's barplot function is used to plot categorical data, where the x-axis represents different categories (Class), and the y-axis represents the corresponding data values. The estimator='mean' argument allows automatic aggregation, providing the average value for each category. This helps in identifying trends and comparisons across different classes efficiently. The rotation of x-axis labels enhances readability.

```
table = pd.crosstab(df['Outcome'], df['Pregnancies'])
print("Contingency Table:\n", table)

table.plot(kind='bar', figsize=(10, 5))
plt.title("Pregnancies vs. Outcome")
plt.xlabel("Pregnancies")
plt.ylabel("Count")
plt.legend(title='Outcome')
plt.show()
```



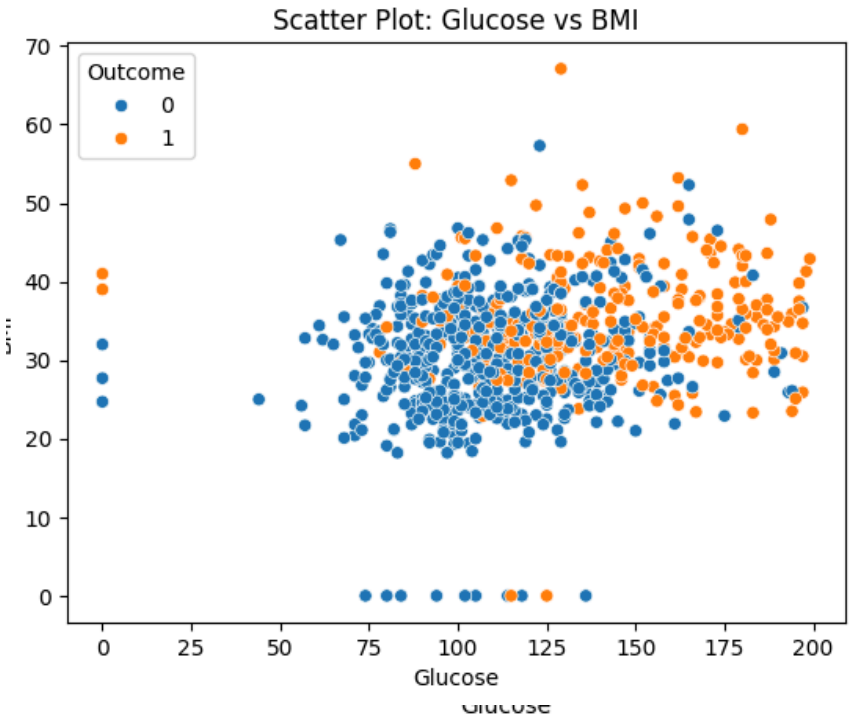
2.**Plot Scatter Plot, Box Plot, and Heatmap**

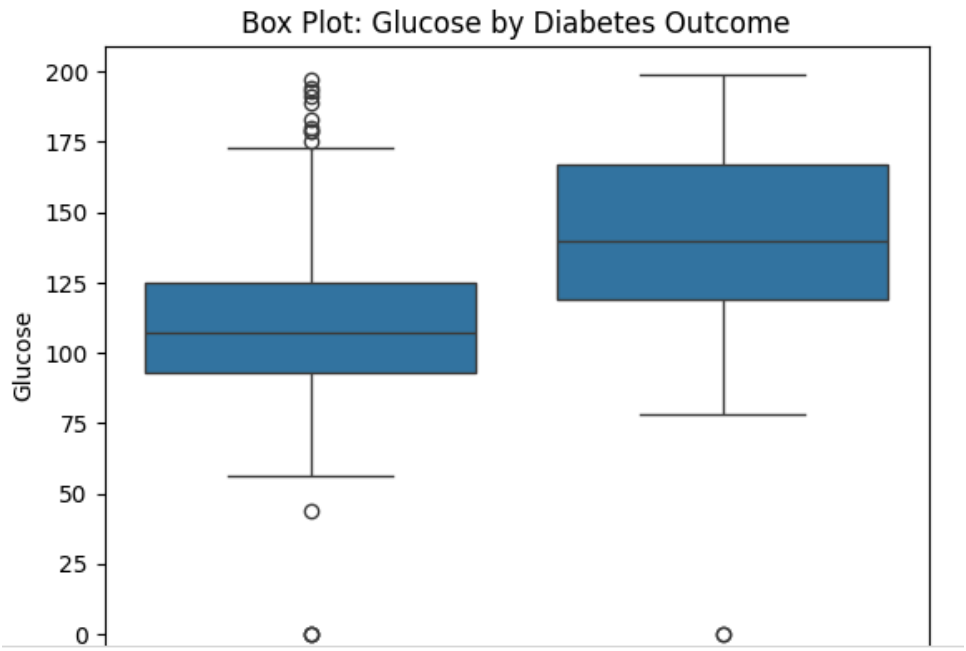
In this part, we have plotted Heatmap and seaborn as they are relevant to our dataset. A heatmap is used to visualize the contingency table, which represents the relationship between two categorical variables, Region and Class. Seaborn's heatmap function is applied to display data distribution using a color gradient (cmap="coolwarm"), where darker shades indicate higher values. The annot=True argument ensures that numerical values are displayed in each cell for clarity. Matplotlib is used to set figure size and labels, enhancing readability. This visualization helps in quickly identifying trends and correlations between categories

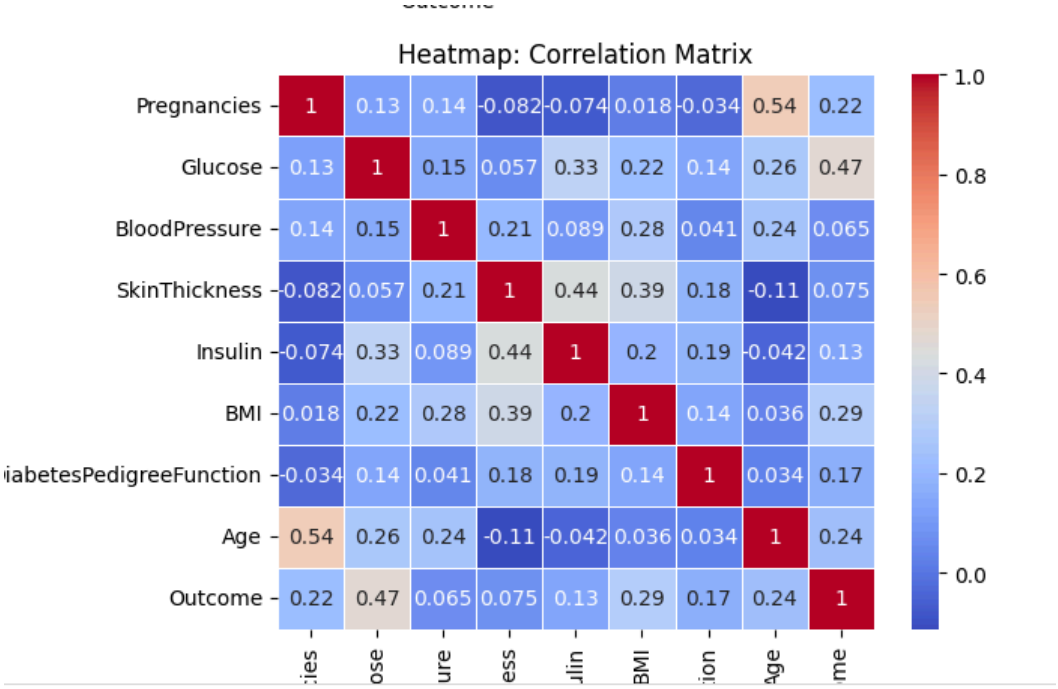
```
sns.scatterplot(x=df['Glucose'], y=df['BMI'], hue=df['Outcome'])
plt.title("Glucose vs. BMI")
plt.show()

sns.boxplot(x=df['Outcome'], y=df['BloodPressure'])
plt.title("Boxplot of Blood Pressure by Outcome")
plt.show()

plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Heatmap")
plt.show()
```


Scatter Plot: Glucose vs BMI
Outcome
0
1
70
60
50
40
30
20
10
0
0
25
50
75
100
125
150
175
200
Glucose

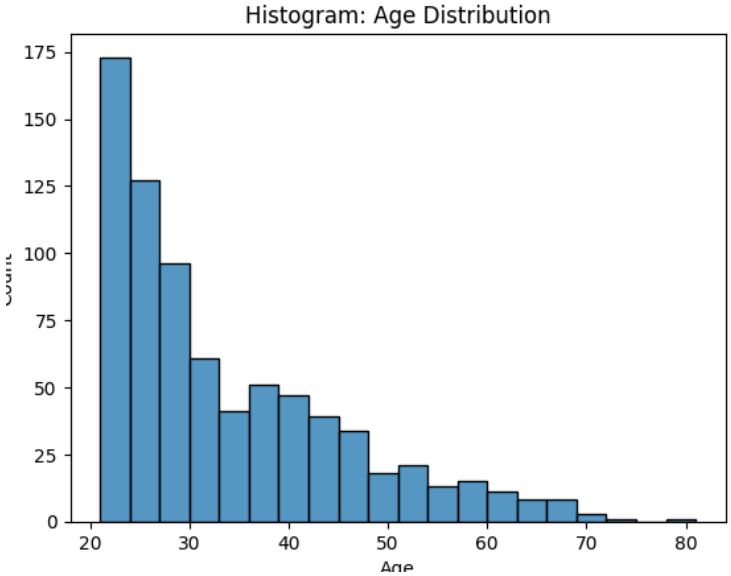

Box Plot: Glucose by Diabetes Outcome
Glucose
200
175
150
125
100
75
50
25
0

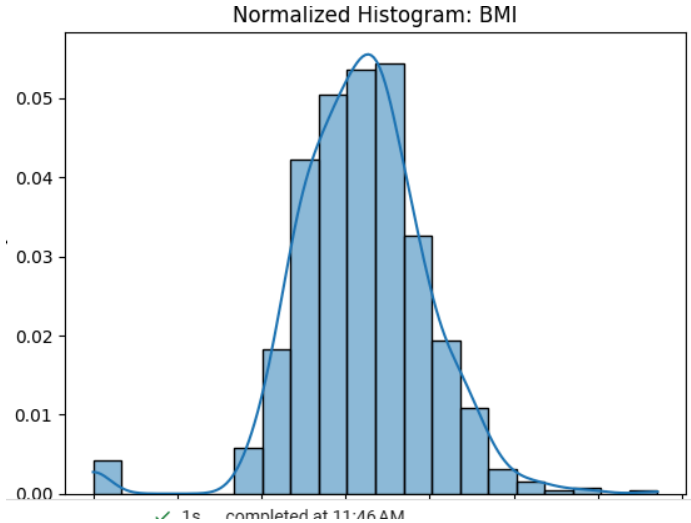


3.**Create Histogram and Normalized Histogram**

A histogram is used to visualize the distribution of Data_Value, showing how frequently different values occur. The sns.histplot function from Seaborn is used with bins=30 to divide data into 30 intervals, providing a detailed view of the distribution. The kde=True parameter adds a Kernel Density Estimate (KDE) curve for a smooth representation. The color is set to blue for better visibility. This helps in understanding data skewness, spread, and common value ranges.

```
plt.hist(df['Age'], bins=20, alpha=0.7, color='blue', edgecolor='black')
plt.title("Age Distribution Histogram")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```





4.Describe What This Graph and Table Indicates

Bar Graph: The bar graph shows the distribution of diabetes outcomes. You can easily see the number of individuals with and without diabetes.

Contingency Table: The table shows the relationship between `Pregnancies` and `Outcome`. It tells you how many people with different numbers of pregnancies have diabetes (1) or not (0).

Scatter Plot: This shows how `Glucose` and `BMI` are related, and whether people with and without diabetes have different patterns in these variables.
Box Plot: The box plot illustrates the distribution of `Glucose` levels for people with diabetes vs. those without it. It helps identify the spread, median, and potential outliers in the glucose levels.
Heatmap: The heatmap shows how correlated the numerical features are with each other. For example, `BMI` might be highly correlated with `Insulin` or `Age`.
Histograms: The regular histogram of `Age` shows the frequency distribution of age in the dataset, while the normalized histogram for `BMI` shows how BMI values are distributed across the population in terms of probability density.

5.**Handle Outlier Using Box Plot and Inter Quartile Range (IQR)**

Outliers in a dataset can significantly impact statistical analysis, making it essential to identify and handle them effectively. A boxplot is a useful visualization tool for detecting outliers, as it highlights the spread of data, median, and interquartile range (IQR). The IQR method is a robust technique for outlier detection, where values falling outside Q1 - 1.5 * IQR and Q3 + 1.5 * IQR are considered potential outliers. By computing Q1 (25th percentile) and Q3 (75th percentile), we determine the IQR and set boundaries to identify extreme values. Outliers can distort measures of central tendency and variability, potentially leading to misleading insights. Once detected, these values can be analyzed to determine if they result from data entry errors, anomalies, or natural variability. Proper handling, such as transformation, capping, or removal, depends on the dataset's context. The combination of a boxplot and IQR analysis ensures a balanced approach to managing outliers.

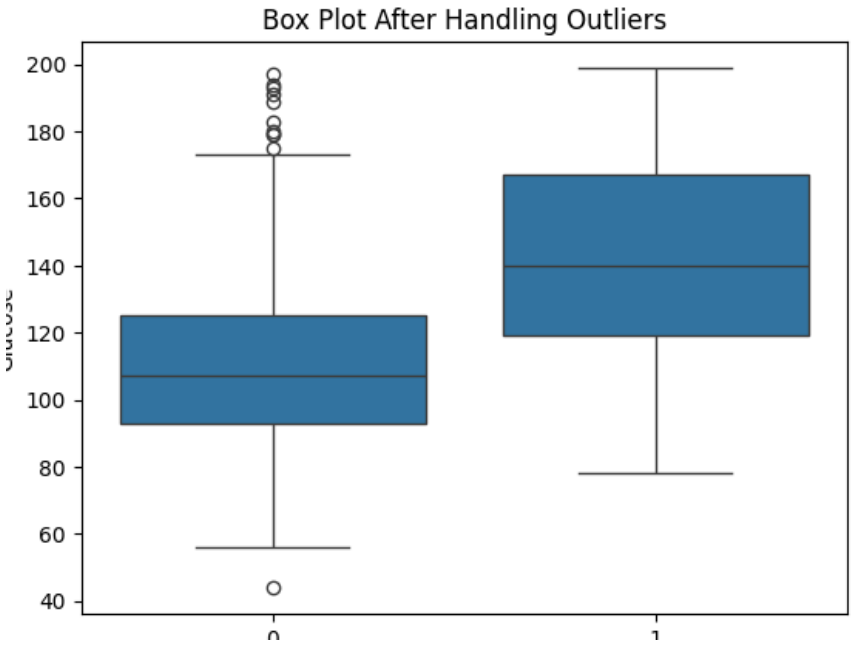
```
Q1 = df.quantile(0.25)
Q3 = df.quantile(0.75)
IQR = Q3 - Q1
```

```
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
```

```
df_no_outliers = df[~((df < lower_bound) | (df > upper_bound)).any(axis=1)]
print("Data shape before outlier removal:", df.shape)
print("Data shape after outlier removal:", df_no_outliers.shape)
```

```
Data shape before outlier removal: (768, 9)
Data shape after outlier removal: (639, 9)
```

```
sns.boxplot(x='Outcome', y='Insulin', data=df_no_outliers)
plt.title('Box Plot of Insulin by Outcome (Outliers Removed)')
plt.show()
```



6.Conclusion: This experiment provided valuable insights into the dataset through various visualizations. The bar graph highlighted variations in Data_Value across different categories, helping identify dominant and underrepresented classes. The contingency table and heatmap effectively showcased regional differences, revealing areas with higher concentrations in specific categories.The histogram and KDE curve indicated a multimodal distribution, suggesting the presence of distinct subgroups within the data. The box plot and IQR method identified several outliers, with extreme values that could potentially skew statistical analysis. Detecting and managing these outliers is crucial to maintaining data accuracy and ensuring reliable conclusions. Overall, this study reinforced the importance of exploratory data analysis (EDA) in understanding data distribution, detecting patterns, and handling anomalies. These visualizations provide a clearer perspective on data trends, aiding in better decision-making and more precise interpretations of the dataset.