EXPLORATORY DATA ANALYSIS OF FRAMINGHAM HEART STUDY DATASET

INTRODUCTION

The Framingham Heart Study dataset is a valuable resource for understanding factors associated with cardiovascular health. In this EDA, we will explore the data, clean it, and generate insights to gain a deeper understanding of the dataset.

- **Objective**: Our primary goal is to investigate the relationship between cholesterol levels and the risk of developing Coronary Heart Disease (CHD) within ten years.
- **Hypothesis**: We aim to test the hypothesis that "high cholesterol is associated with a higher risk of developing CHD within ten years."

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Loading the dataset

```
fram = pd.read_csv("/content/framingham.csv")
```

DATA EXPLORATION

```
# to check the shape of the dataset
fram.shape
```

```
(4240, 16)
```

We can see that the dataset consists of 4240 rows and 16 columns. Now, let's see the first five rows of our dataset

```
fram.head()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BM
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.9
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.7
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.3
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.5
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.1

```
fram.describe()
```

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	t
count	4240.000000	4240.000000	4135.000000	4240.000000	4211.000000	4187.000000	4240.000000	4240.000000	4240.000000	419
mean	0.429245	49.580189	1.979444	0.494104	9.005937	0.029615	0.005896	0.310613	0.025708	236
std	0.495027	8.572942	1.019791	0.500024	11.922462	0.169544	0.076569	0.462799	0.158280	44
min	0.000000	32.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	107
25%	0.000000	42.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	206
50%	0.000000	49.000000	2.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	234
75%	1.000000	56.000000	3.000000	1.000000	20.000000	0.000000	0.000000	1.000000	0.000000	263
max	1.000000	70.000000	4.000000	1.000000	70.000000	1.000000	1.000000	1.000000	1.000000	696

```
# Displaying the columns in the dataset
print("\nColumns in the dataset:")
print(fram.columns)
```

```
Columns in the dataset:
Index(['male', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'TenYearCHD'],
      dtype='object')
```

```
fram.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   male             4240 non-null   int64  
 1   age              4240 non-null   int64  
 2   education        4135 non-null   float64
 3   currentSmoker    4240 non-null   int64  
 4   cigsPerDay       4211 non-null   float64
 5   BPMeds           4187 non-null   float64
 6   prevalentStroke  4240 non-null   int64  
 7   prevalentHyp     4240 non-null   int64  
 8   diabetes         4240 non-null   int64  
 9   totChol          4190 non-null   float64
 10  sysBP            4240 non-null   float64
 11  diaBP            4240 non-null   float64
 12  BMI              4221 non-null   float64
 13  heartRate        4239 non-null   float64
 14  glucose          3852 non-null   float64
 15  TenYearCHD       4240 non-null   int64  
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

```
# Checking for missing values
print("\nMissing Values:")
print(fram.isnull().sum())
```

```
Missing Values:
male                 0
age                  0
education          105
currentSmoker        0
cigsPerDay          29
BPMeds              53
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             50
sysBP                0
diaBP                0
BMI                 19
heartRate            1
glucose            388
TenYearCHD           0
dtype: int64
```

From the above cell, we noticed that:

The columns education, cigsPerDay, BPMeds,totChol, sysBP, diaBP, BMI, heartRate and glucose has missing values. Most of the columns are not of their respective datatype they should be.

So, now we will start with the data cleaning and transformation phase.

DATA CLEANING AND HANDLING MISSING VALUES

The first step in any data analysis is to ensure the dataset is clean and ready for exploration. We start by checking for missing values and making necessary adjustments:

Checking for missing values

```
# Check for missing values
missing_values = fram.isnull().sum()
```

Now, we can check whether there are null values in the dataframe.

```
# checking for null values
fram.isnull().mean()*100
```

```
male               0.000000
age                0.000000
education          2.476415
currentSmoker      0.000000
cigsPerDay         0.683962
BPMeds             1.250000
prevalentStroke    0.000000
prevalentHyp       0.000000
diabetes           0.000000
totChol            1.179245
sysBP              0.000000
diaBP              0.000000
BMI                0.448113
heartRate          0.023585
glucose            9.150943
TenYearCHD         0.000000
dtype: float64
```

```
print("Missing values in education column: ",fram.education.isna().sum())
print("Missing values in cigsPerDay column: ",fram.cigsPerDay.isna().sum())
print("Missing values in BPMeds column: ",fram.BPMeds.isna().sum())
print("Missing values in totChol column: ",fram.totChol.isna().sum())
print("Missing values in BMI column: ",fram.BMI.isna().sum())
print("Missing values in heartRate column: ",fram.heartRate.isna().sum())
```

```
Missing values in education column:  105
Missing values in cigsPerDay column:  29
Missing values in BPMeds column:  53
Missing values in totChol column:  50
Missing values in BMI column:  19
Missing values in heartRate column:  1
```

We examined the dataset for missing values and found that some columns have missing data. We decided to add values to address the missing data.

So, we calcuated the total missing values of each of the column. Now, let us fill up these missing values. We are making these assumptions to fill up the missing values:

1. Missing value in education means that education level might be similar to the most frequent value therfore we will use mode.
2. Assuming that missing values in CigsPerDay Column represent non-smokers (0 cigarettes per day) so we will assign it with a value of 0.
3. Missing values in BPMeds Column indicate no blood pressure medications so we will assign it with a value of 0.
4. Assuming that missing values in TotChol Column are replaced with the mean cholesterol level.
5. Assuming that missing values in BMI Column are replaced with the mean BMI.
6. Missing values in HeartRate Column are replaced with the mean heart rate.

```
# Summary statistics for numeric columns
print("\nSummary Statistics:")
print(fram.describe())
```

```
Summary Statistics:
              male          age    education  currentSmoker   cigsPerDay  \
count  4240.000000  4240.000000  4135.000000    4240.000000  4211.000000
mean      0.429245    49.580189     1.979444       0.494104     9.005937
std       0.495027     8.572942     1.019791       0.500024    11.922462
min       0.000000    32.000000     1.000000       0.000000     0.000000
25%       0.000000    42.000000     1.000000       0.000000     0.000000
50%       0.000000    49.000000     2.000000       0.000000     0.000000
75%       1.000000    56.000000     3.000000       1.000000    20.000000
max       1.000000    70.000000     4.000000       1.000000    70.000000
```

```
            BPMeds  prevalentStroke  prevalentHyp     diabetes      totChol  \
count  4187.000000      4240.000000   4240.000000  4240.000000  4190.000000
mean      0.029615         0.005896      0.310613     0.025708   236.699523
std       0.169544         0.076569      0.462799     0.158280    44.591284
min       0.000000         0.000000      0.000000     0.000000   107.000000
25%       0.000000         0.000000      0.000000     0.000000   206.000000
50%       0.000000         0.000000      0.000000     0.000000   234.000000
75%       0.000000         0.000000      1.000000     0.000000   263.000000
max       1.000000         1.000000      1.000000     1.000000   696.000000

             sysBP        diaBP          BMI    heartRate      glucose  \
count  4240.000000  4240.000000  4221.000000  4239.000000  3852.000000
mean    132.354599    82.897759    25.800801    75.878981    81.963655
std      22.033300    11.910394     4.079840    12.025348    23.954335
min      83.500000    48.000000    15.540000    44.000000    40.000000
25%     117.000000    75.000000    23.070000    68.000000    71.000000
50%     128.000000    82.000000    25.400000    75.000000    78.000000
75%     144.000000    90.000000    28.040000    83.000000    87.000000
max     295.000000   142.500000    56.800000   143.000000   394.000000

         TenYearCHD
count  4240.000000
mean      0.151887
std       0.358953
min       0.000000
25%       0.000000
50%       0.000000
75%       0.000000
max       1.000000
```

We calculated basic statistics for numeric columns, including mean, standard deviation, minimum, and maximum values. This provides a quick overview of the data's central tendencies.

```
#checking the education column
fram.education[fram.education==1.979444]
```

```
Series([], Name: education, dtype: float64)
```

```
#checking the totChol column
fram.totChol[fram.totChol==236.699523 ]
```

```
Series([], Name: totChol, dtype: float64)
```

```
#checking the BMI column
fram.BMI[fram.BMI==25.800801]
```

```
Series([], Name: BMI, dtype: float64)
```

```
#checking the heartRate column
fram.heartRate[fram.heartRate==75.878981]
```

```
Series([], Name: heartRate, dtype: float64)
```

```
fram.education.fillna(0,inplace=True) # filling the missing values of education with 0
fram.cigsPerDay.fillna(0,inplace=True) #filling the missing value with 0( for no company sponsorship)
fram.BPMeds.fillna(0,inplace=True) #filling the missing value with 0 (for any null values)
fram.totChol.fillna(236.699523,inplace=True)
fram.BMI.fillna(25.800801,inplace=True)
fram.heartRate.fillna(75.878981 ,inplace=True)
fram.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4240 entries, 0 to 4239
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   male             4240 non-null   int64
 1   age              4240 non-null   int64
 2   education        4240 non-null   float64
 3   currentSmoker    4240 non-null   int64
 4   cigsPerDay       4240 non-null   float64
 5   BPMeds           4240 non-null   float64
 6   prevalentStroke  4240 non-null   int64
 7   prevalentHyp     4240 non-null   int64
 8   diabetes         4240 non-null   int64
 9   totChol          4240 non-null   float64
 10  sysBP            4240 non-null   float64
```

```
 11  diaBP           4240 non-null   float64
 12  BMI             4240 non-null   float64
 13  heartRate       4240 non-null   float64
 14  glucose         3852 non-null   float64
 15  TenYearCHD      4240 non-null   int64
dtypes: float64(9), int64(7)
memory usage: 530.1 KB
```

We are done with replacing the missing values for education, cigsPerDay, BPMeds, totChol, BMI, and heartRate.

```
fram.drop_duplicates(inplace=True)
```

```
data_types = fram.dtypes
```

```
# Save the cleaned dataset to a new CSV file
fram.to_csv('cleaned_framingham.csv', index=False)

print("Cleaned dataset saved as 'cleaned_framingham.csv'")
```

```
Cleaned dataset saved as 'cleaned_framingham.csv'
```

```
# Loading the cleaned dataset
fram = pd.read_csv('cleaned_framingham.csv')
```

DATA AGGREGATION

```
# Defining the null hypothesis
null_hypothesis = "There is no significant difference in the ten-year CHD risk between individuals with high cholesterol and thc
```

```
# Data Aggregation and Reporting Summaries
# Categorize cholesterol levels into 'Normal' and 'High'
df['cholesterol_category'] = pd.cut(df['totChol'], bins=[0, 200, float('inf')], labels=['Normal', 'High'])
```

```
# Calculate the proportion of CHD cases among different cholesterol levels
chd_proportion = df.groupby('cholesterol_category')['TenYearCHD'].mean()
print("Proportion of CHD Cases among Different Cholesterol Levels:")
print(chd_proportion)

# Compute summary statistics for key variables
high_cholesterol_summary = df[df['cholesterol_category'] == 'High'].describe()
normal_cholesterol_summary = df[df['cholesterol_category'] == 'Normal'].describe()
```

```
Proportion of CHD Cases among Different Cholesterol Levels:
cholesterol_category
Normal    0.104848
High      0.164330
Name: TenYearCHD, dtype: float64
```

The proportion of CHD cases among different cholesterol levels (Normal and High) that you've provided suggests that a higher proportion of individuals with high cholesterol (High cholesterol category) experience CHD compared to those with normal cholesterol (Normal cholesterol category).

For the "Normal" cholesterol category, approximately 10.49% of individuals experienced CHD. For the "High" cholesterol category, approximately 16.43% of individuals experienced CHD. This information indicates that there is a notable difference in the risk of CHD between these two cholesterol categories. High cholesterol appears to be associated with a higher risk of developing CHD compared to normal cholesterol levels. These proportions can be used to better understand the relationship between cholesterol levels and CHD in your dataset.

We can further understand this theory by visualizing them.

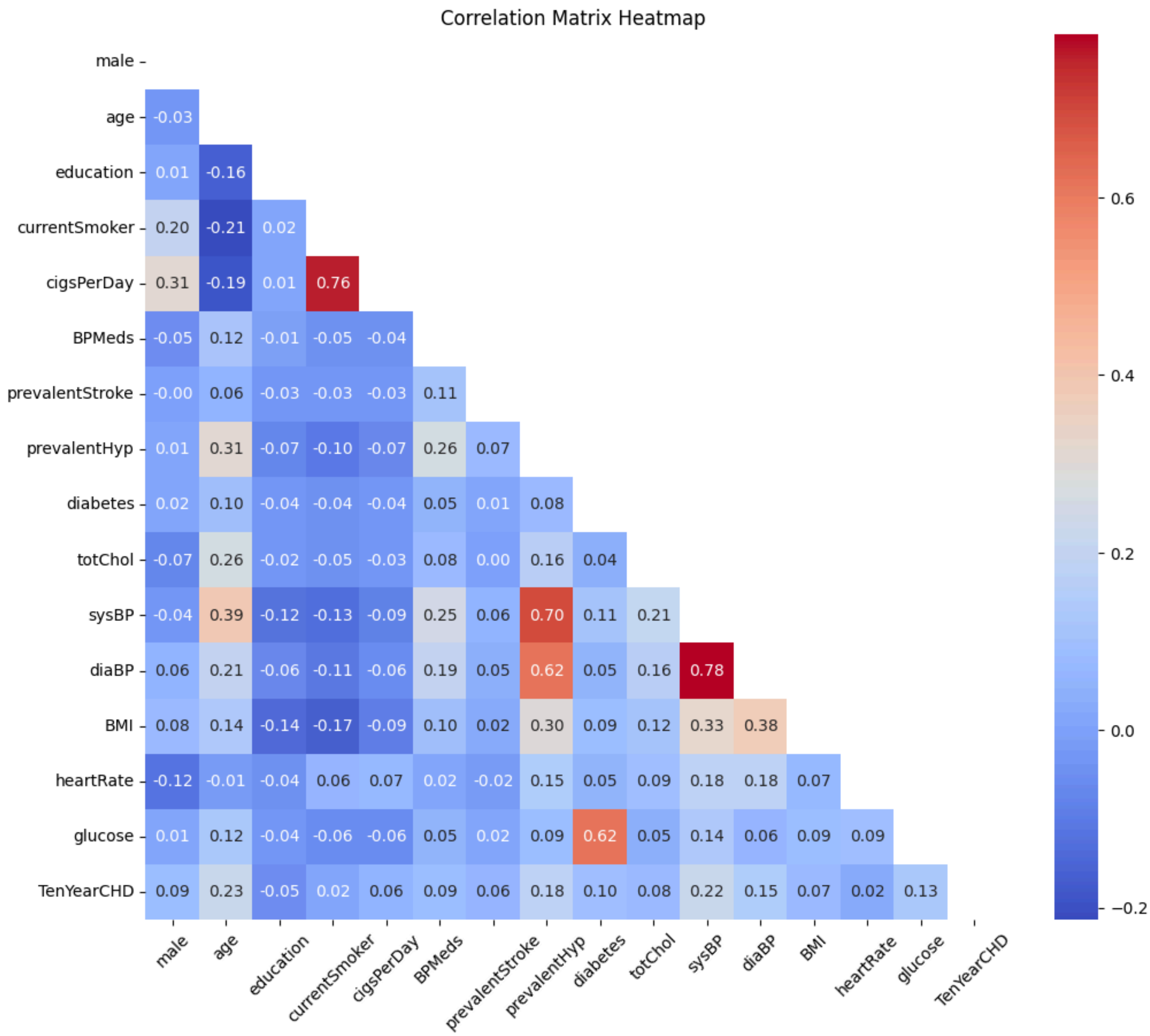
DATA VISUALIZATION

Visualizations are powerful tools to uncover patterns and relationships in the data.

Correlation matrix

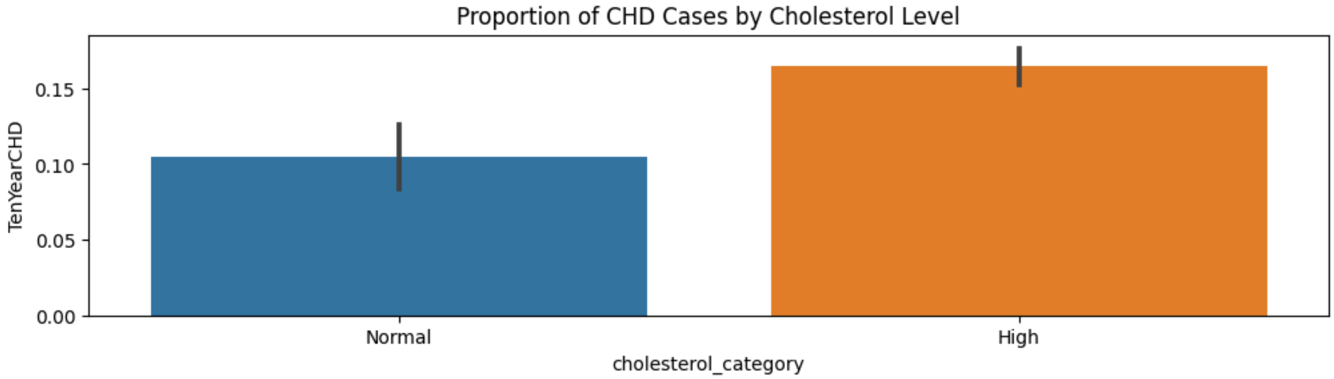
The correlation matrix offers insights into relationships between variables, the identification of risk factors, and guidance for interventions. It's important in understanding how different factors are interconnected and their impact on cardiovascular health. These insights can be valuable for public health initiatives, further research, and predictive modeling.

```
# Calculating the correlation matrix with numeric columns only
correlation_matrix = df.corr(numeric_only=True)

# Create a mask for the upper triangle to remove duplicate values
mask = np.triu(np.ones_like(correlation_matrix, dtype=bool))

plt.figure(figsize=(12, 10))

# Creating a heatmap
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap="coolwarm", mask=mask)
plt.title("Correlation Matrix Heatmap")
plt.xticks(rotation=45)  # Rotate x-axis labels for better readability
plt.show()
```



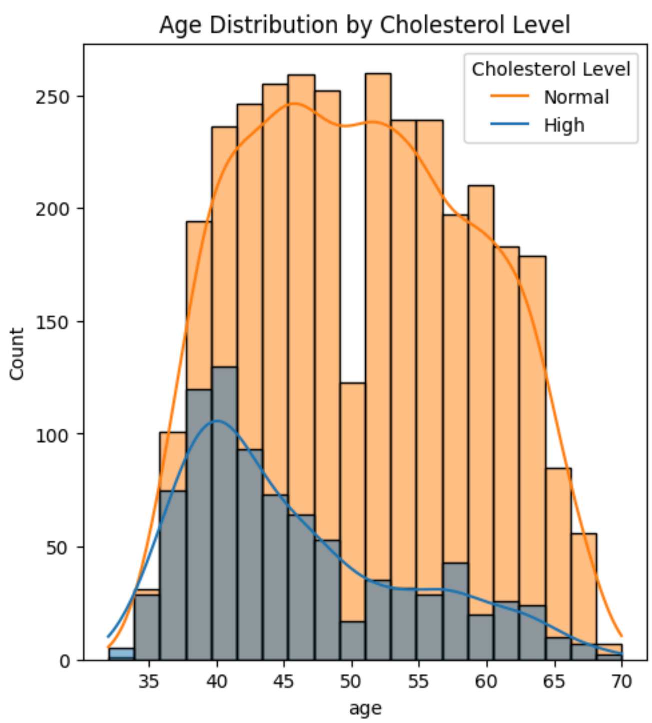

```
# Loading the cleaned dataset
df = pd.read_csv('cleaned_framingham.csv')
# Creating a 'cholesterol_category' based on cholesterol levels
df['cholesterol_category'] = pd.cut(df['totChol'], bins=[0, 200, df['totChol'].max()], labels=['Normal', 'High'], include_lowest

# Proportion of CHD Cases by Cholesterol Level
plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 2)
sns.barplot(x='cholesterol_category', y='TenYearCHD', data=df)
plt.title("Proportion of CHD Cases by Cholesterol Level")
```

```
Text(0.5, 1.0, 'Proportion of CHD Cases by Cholesterol Level')
```



The age distribution in the Framingham Heart Study dataset is a crucial component for understanding the cardiovascular health of the participants. It provides insights into demographic profiles, age-related trends, risk assessment, and potential areas for targeted interventions and research. The age distribution is fundamental for investigating the impact of age on cardiovascular health and guiding public health strategies.

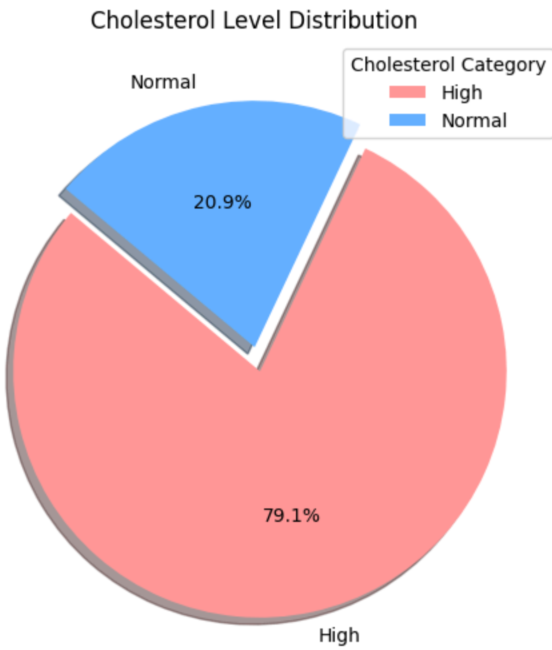
```
#Creating a Histogram by using the Age Distribution by Cholesterol Level

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(df, x='age', hue='cholesterol_category', bins=20, kde=True)
plt.title("Age Distribution by Cholesterol Level")
plt.legend(title='Cholesterol Level', labels=['Normal', 'High'], loc='upper right')

plt.show()
```



Histogram demonstrates the age distribution for both groups, showing similarities but hinting at a slight density difference among older individuals in the 'High' cholesterol group. We can observe that the age distribution does not significantly differ between the 'High' and 'Normal' cholesterol groups, suggesting that age may not be the primary factor influencing CHD risk in this dataset.

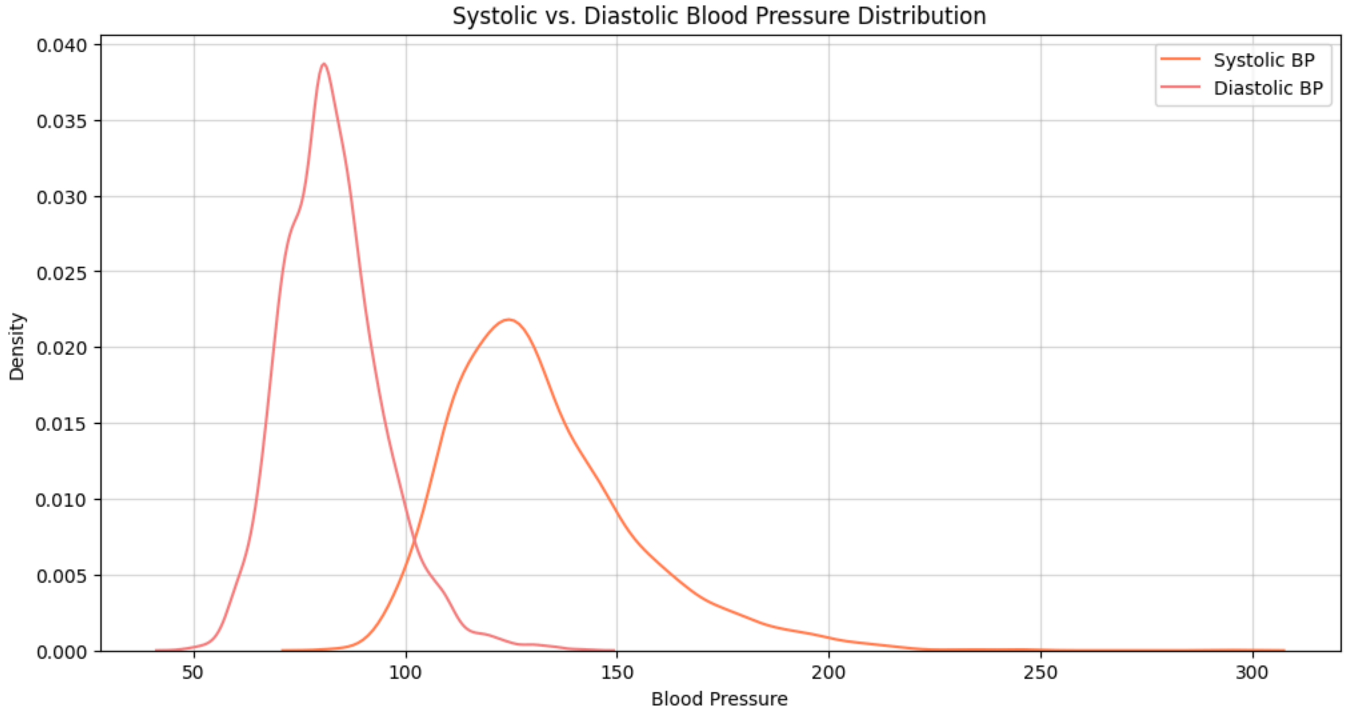
```
#Creating a pie chart to understand the Cholestrol Level Distribution
cholesterol_counts = df['cholesterol_category'].value_counts()
labels = cholesterol_counts.index
sizes = cholesterol_counts
explode = (0.1, 0)  # Exploding the "High" cholesterol slice
```

```
colors = ['#FF9999', '#66B2FF']  # Took reference from matplot libraries
plt.figure(figsize=(6, 6))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', explode=explode, colors=colors, shadow=True, startangle=140)
plt.legend(labels, title="Cholesterol Category", loc="upper right")
plt.title("Cholesterol Level Distribution")

plt.show()
```



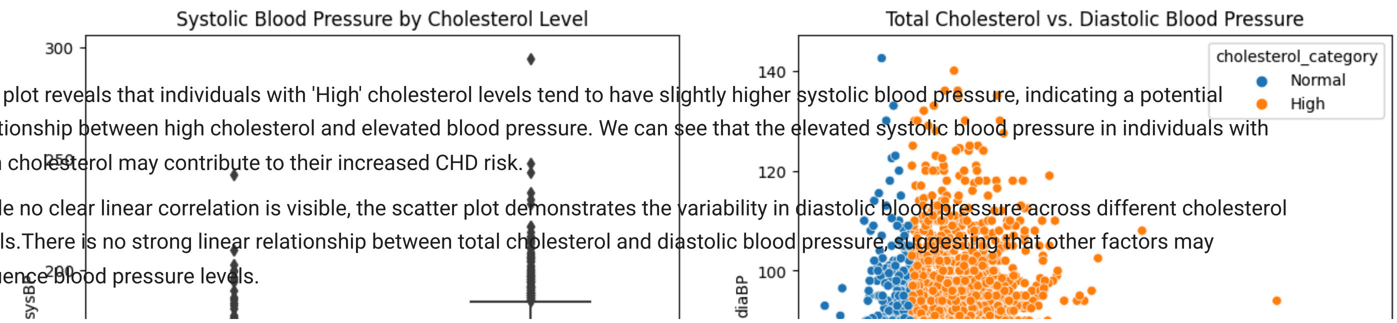
This pie chart provides an overview of the distribution of 'High' and 'Normal' cholesterol levels in the dataset, highlighting that the majority of individuals have 'High' cholesterol levels.

```
#Blood Pressure Distribution (Systolic vs Diastolic)
plt.figure(figsize=(12, 6))
sns.kdeplot(data=df, x='sysBP', label='Systolic BP', color='coral')
sns.kdeplot(data=df, x='diaBP', label='Diastolic BP', color='lightcoral')
plt.title('Systolic vs. Diastolic Blood Pressure Distribution')
plt.xlabel('Blood Pressure')
plt.ylabel('Density')
plt.legend()
plt.grid(alpha=0.5)
plt.show()
```



Analyzing the blood pressure distribution in the dataset is crucial for understanding the prevalence of hypertension and its relationship with cardiovascular health. It can guide public health strategies, identify at-risk populations, and offer insights into the impact of age, gender, and other factors on blood pressure.

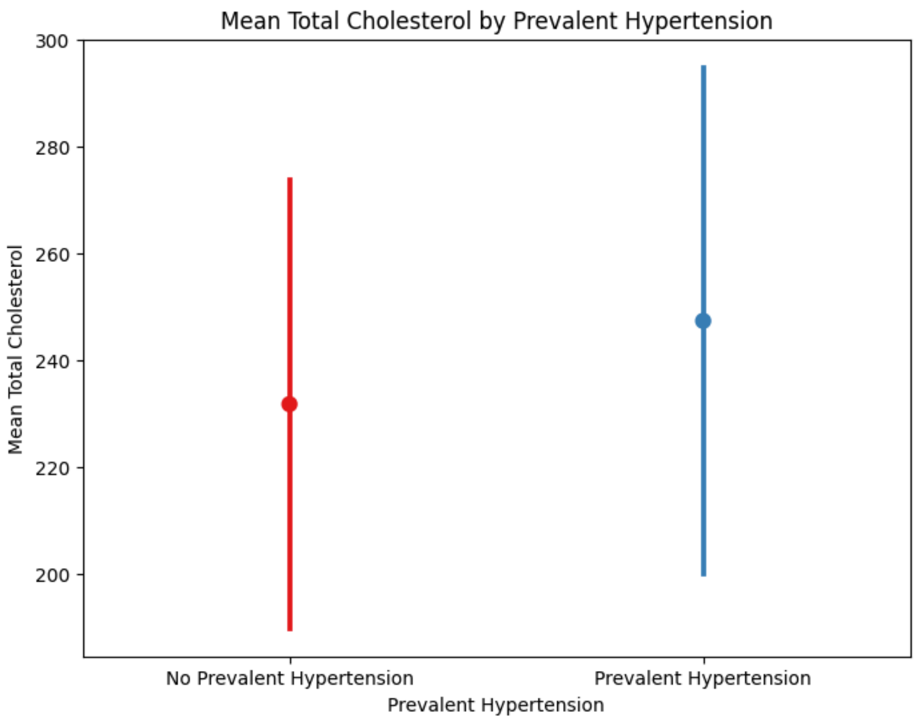
```
plt.figure(figsize=(15, 6))

# Creating a Box Plot for Systolic Blood Pressure by Cholesterol Level
plt.subplot(1, 2, 1)
sns.boxplot(x='cholesterol_category', y='sysBP', data=df)
plt.title("Systolic Blood Pressure by Cholesterol Level")

# Creating a Scatter Plot for Total Cholesterol vs. Diastolic Blood Pressure
plt.subplot(1, 2, 2)
sns.scatterplot(data=df, x='totChol', y='diaBP', hue='cholesterol_category')
plt.title("Total Cholesterol vs. Diastolic Blood Pressure")

plt.show()
```



Box plot reveals that individuals with 'High' cholesterol levels tend to have slightly higher systolic blood pressure, indicating a potential relationship between high cholesterol and elevated blood pressure. We can see that the elevated systolic blood pressure in individuals with high cholesterol may contribute to their increased CHD risk.

While no clear linear correlation is visible, the scatter plot demonstrates the variability in diastolic blood pressure across different cholesterol levels.There is no strong linear relationship between total cholesterol and diastolic blood pressure, suggesting that other factors may influence blood pressure levels.

```
# Creating a Point plot for understanding the total cholestrol by prevalent Hypertension
plt.figure(figsize=(8, 6))
sns.pointplot(x='prevalentHyp', y='totChol', data=df, palette='Set1', errorbar='sd')

# Add labels to the x-axis
plt.xticks([0, 1], ['No Prevalent Hypertension', 'Prevalent Hypertension'])

plt.title('Mean Total Cholesterol by Prevalent Hypertension')
plt.xlabel('Prevalent Hypertension')
plt.ylabel('Mean Total Cholesterol')
plt.show()
```



The point graph clearly shows a difference in mean total cholesterol levels between individuals with prevalent hypertension and those