```
!pip install faker
```

```
Collecting faker
  Downloading Faker-23.1.0-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 1.9 MB/s eta 0:00:00
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)
Installing collected packages: faker
Successfully installed faker-23.1.0
```

```
import random
import pandas as pd
from faker import Faker

fake = Faker()

# Generate data for 1000 patients
data = []
for _ in range(1000):
    name = fake.name()
    age = random.randint(1, 90) if random.random() < 0.95 else None  # Introduce missing values in age
    if age is not None:
        if age <= 18:
            age_group = 'Child' if age <= 12 else 'Teenager'
        else:
            age_group = 'Adult'
    else:
        age_group = 'Unknown'

    gender = random.choice(['Male', 'Female'])
    race = random.choice(['Asian', 'Black', 'Hispanic', 'White', 'Other'])
    blood_group = random.choice(['A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'])
    temperature = round(random.uniform(97.0, 99.0), 1)
    blood_pressure_sys = random.randint(90, 180)  # Allow for outliers
    blood_pressure_dia = random.randint(60, 120)  # Allow for outliers
    height = round(random.uniform(120, 200), 2)  # in centimeters
    weight = round(random.uniform(40, 150), 2)   # in kilograms

    # Simulate common health concerns based on age and gender
    common_health_concerns = []
    if age_group == 'Child':
        common_health_concerns.append('Vaccination')
        common_health_concerns.append('Growth and development')
    elif age_group == 'Teenager':
        common_health_concerns.append('Acne and skin issues')
        common_health_concerns.append('Mental health challenges')
        common_health_concerns.append('Sexual health education')
    elif age_group == 'Adult':
        common_health_concerns.append('Chronic diseases management')
        common_health_concerns.append('Heart health monitoring')
        common_health_concerns.append('Diet and exercise counseling')

    # Simulate additional health concerns
    additional_health_concerns = [fake.random.choice(['Asthma', 'Diabetes', 'Hypertension', 'Obesity', 'Allergies', 'Anxiety', 'Depress

    # Combine common and additional health concerns
    health_concerns = common_health_concerns + additional_health_concerns

    # Determine if the patient needs medical attention based on health concerns and vital signs
    if any(condition in health_concerns for condition in ['Hypertension', 'Diabetes', 'Asthma', 'Allergies', 'Depression']):
        needs_medical_attention = 1
    elif blood_pressure_sys > 140 or blood_pressure_dia > 90 or weight > 100:
        needs_medical_attention = 1
    else:
        needs_medical_attention = 0

    # Simulate patients visiting for general checkups
    if random.random() < 0.2:  # 20% chance of visiting for a general checkup
        health_concerns = 'General Checkup'
        needs_medical_attention = 0

    data.append({
        'Name': name,
        'Age': age,
        'Age Group': age_group,
        'Gender': gender,
        'Race': race,
        'Blood Group': blood_group,
        'Temperature': temperature,
        'Blood Pressure (Systolic)': blood_pressure_sys,
        'Blood Pressure (Diastolic)': blood_pressure_dia,
        'Height': height,
        'Weight': weight,
        'Health Concerns': ', '.join(health_concerns) if not isinstance(health_concerns, str) else health_concerns,  # Convert list to
        'Needs Medical Attention': needs_medical_attention
    })

# Create a DataFrame
df = pd.DataFrame(data)

# Save the dataset to a CSV file
```

```
df.to_csv('medical_data_v3.csv', index=False)
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
%cd Healthcare
```

```
[Errno 2] No such file or directory: 'Healthcare'
/content
```

```
%ls
```

```
drive/  medical_data_v2.csv  medical_data_v3.csv  newmedical_data.csv  sample_data/
```

```
import pandas as pd

# Assuming your dataset is stored in a CSV file named 'medical_data.csv'
# Replace 'medical_data.csv' with the actual file path if it's located in a different directory
file_path = 'medical_data_v3.csv'

# Read the CSV file into a DataFrame
df = pd.read_csv(file_path)

# Display the first few rows of the DataFrame to verify the import
print("First few rows of the dataset:")
print(df.head())
```

```
First few rows of the dataset:
                  Name   Age Age Group  Gender      Race Blood Group  \
0          Kevin Wong  29.0     Adult  Female     Black         AB+
1    Mr. David Meyers  84.0     Adult    Male     White          A-
2  Robert Sanchez DDS  26.0     Adult    Male  Hispanic          A+
3         Brent Moore  42.0     Adult    Male     Black          B+
4      Alicia Kennedy  24.0     Adult    Male     Black          A+

   Temperature  Blood Pressure (Systolic)  Blood Pressure (Diastolic)  Height  \
0         97.9                        178                          84  183.19
1         98.1                        130                          88  170.96
2         97.9                        121                         101  146.33
3         98.1                        169                         112  197.98
4         98.0                        100                          61  149.24

   Weight                                  Health Concerns  \
0   73.15  Chronic diseases management, Heart health moni...
1   92.36  Chronic diseases management, Heart health moni...
2   77.41  Chronic diseases management, Heart health moni...
3  113.40  Chronic diseases management, Heart health moni...
4   79.90  Chronic diseases management, Heart health moni...

   Needs Medical Attention
0                        1
1                        1
2                        1
3                        1
4                        1
```

```
df
```

	Name	Age	Age Group	Gender	Race	Blood Group	Temperature	Blood Pressure (Systolic)	B Pres (Diasto
0	Kevin Wong	29.0	Adult	Female	Black	AB+	97.9	178	
1	Mr. David Meyers	84.0	Adult	Male	White	A-	98.1	130	
2	Robert Sanchez DDS	26.0	Adult	Male	Hispanic	A+	97.9	121	
3	Brent Moore	42.0	Adult	Male	Black	B+	98.1	169	
4	Alicia Kennedy	24.0	Adult	Male	Black	A+	98.0	100	
...	...	...	...	...	...	...	...	...	
995	Stacy Gallegos	36.0	Adult	Female	Black	A+	97.9	159	
996	Natalie Curtis	15.0	Teenager	Female	Asian	AB+	98.0	97	
997	Taylor Meyer	21.0	Adult	Male	Hispanic	A+	97.6	119	

```
import pandas as pd

breakdown_data = {
    'Column': ['Name', 'Age', 'Age Group', 'Gender', 'Race', 'Blood Group', 'Temperature', 'Blood Pressure (Systolic)',
               'Blood Pressure (Diastolic)', 'Height', 'Weight', 'Health Concerns', 'Needs Medical Attention'],
    'Data Type': ['Nominal', 'Quantitative, Discrete', 'Nominal', 'Nominal', 'Nominal', 'Nominal', 'Continuous', 'Continuous',
                  'Continuous', 'Continuous', 'Continuous', 'Nominal', 'Binary']
}

breakdown_df = pd.DataFrame(breakdown_data)

# Display the breakdown DataFrame
print("Breakdown of Dataset:")
print(breakdown_df)
```

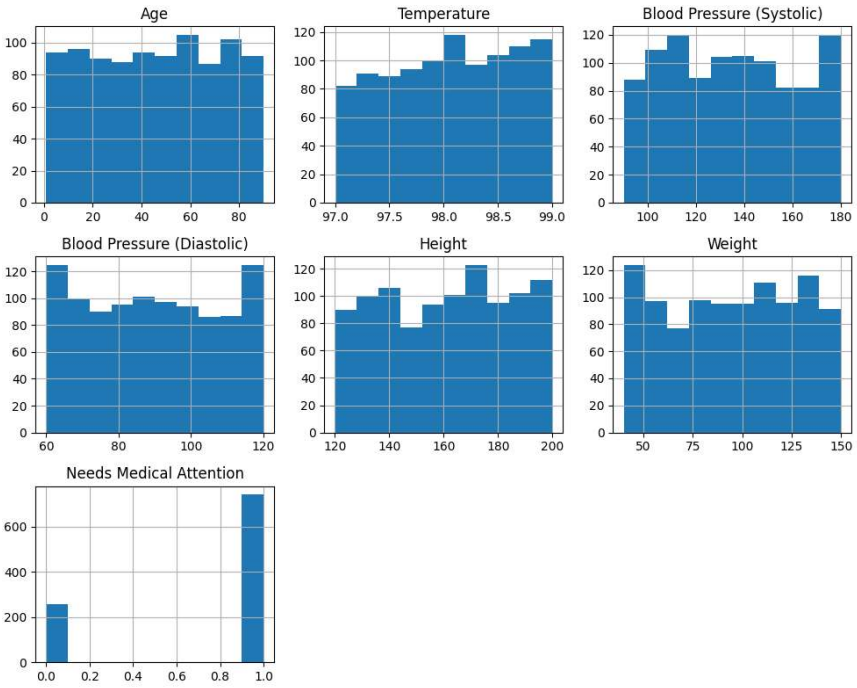
```
Breakdown of Dataset:
                         Column              Data Type
0                          Name                Nominal
1                           Age  Quantitative, Discrete
2                     Age Group                Nominal
3                        Gender                Nominal
4                          Race                Nominal
5                   Blood Group                Nominal
6                   Temperature             Continuous
7    Blood Pressure (Systolic)              Continuous
8   Blood Pressure (Diastolic)              Continuous
9                        Height             Continuous
10                       Weight             Continuous
11             Health Concerns                Nominal
12     Needs Medical Attention                 Binary
```

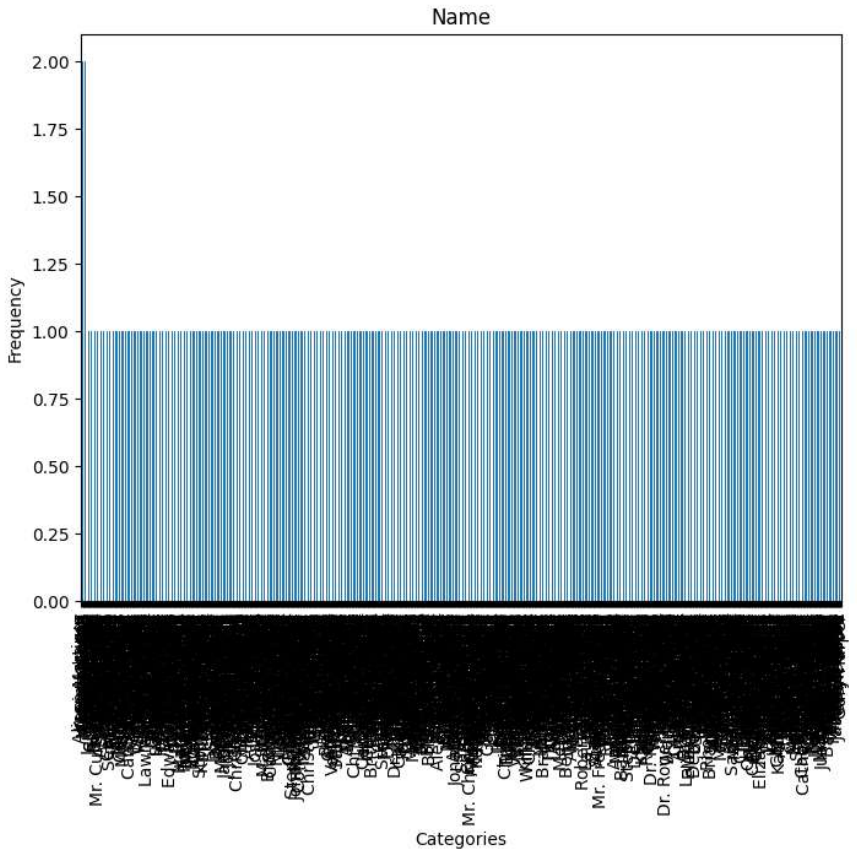
```
print (df)
```

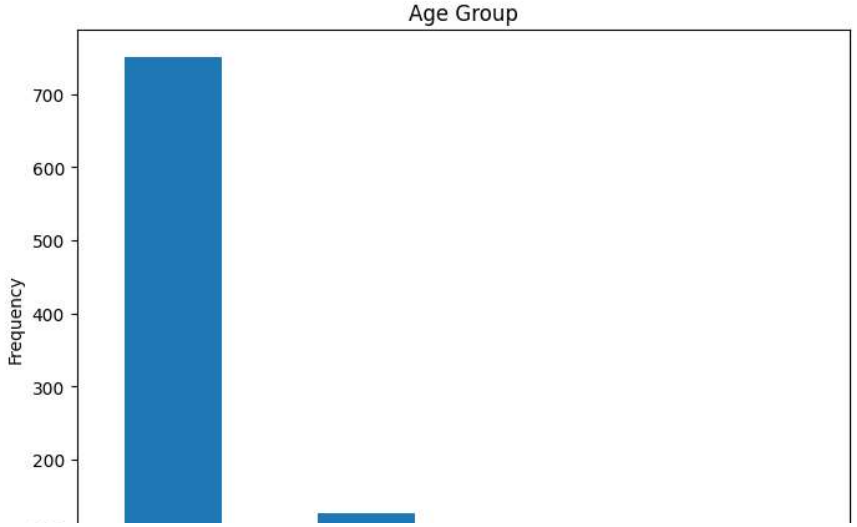
```
                  Name   Age Age Group  Gender      Race Blood Group  \
0           Kevin Wong  29.0     Adult  Female     Black         AB+
```

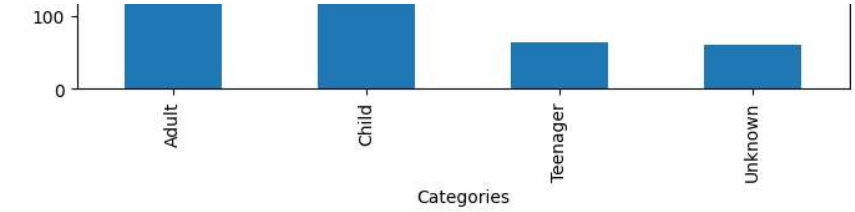
```
1      Mr. David Meyers  84.0     Adult    Male     White          A-
2    Robert Sanchez DDS  26.0     Adult    Male  Hispanic          A+
3           Brent Moore  42.0     Adult    Male     Black          B+
4        Alicia Kennedy  24.0     Adult    Male     Black          A+
..                  ...   ...       ...     ...       ...         ...
995      Stacy Gallegos  36.0     Adult  Female     Black          A+
996      Natalie Curtis  15.0  Teenager  Female     Asian         AB+
997        Taylor Meyer  21.0     Adult    Male  Hispanic          A+
998        Erin Morales  89.0     Adult    Male  Hispanic          A+
999         Gary Harper   1.0     Child    Male     Asian          B+

     Temperature  Blood Pressure (Systolic)  Blood Pressure (Diastolic)  \
0           97.9                        178                          84
1           98.1                        130                          88
2           97.9                        121                         101
3           98.1                        169                         112
4           98.0                        100                          61
..           ...                        ...                         ...
995         97.9                        159                          68
996         98.0                         97                          69
997         97.6                        119                         112
998         98.3                         93                          84
999         98.0                        128                          81

     Height  Weight                                    Health Concerns  \
0    183.19   73.15  Chronic diseases management, Heart health moni...
1    170.96   92.36  Chronic diseases management, Heart health moni...
2    146.33   77.41  Chronic diseases management, Heart health moni...
3    197.98  113.40  Chronic diseases management, Heart health moni...
4    149.24   79.90  Chronic diseases management, Heart health moni...
..      ...     ...                                               ...
995  162.34  136.16  Chronic diseases management, Heart health moni...
996  182.38   55.07  Acne and skin issues, Mental health challenges...
997  185.71  117.14                                    General Checkup
998  144.21  109.79                                    General Checkup
999  194.56   77.49                                    General Checkup

     Needs Medical Attention
0                          1
1                          1
2                          1
3                          1
4                          1
..                       ...
995                        1
996                        0
997                        0
998                        0
999                        0

[1000 rows x 13 columns]
```

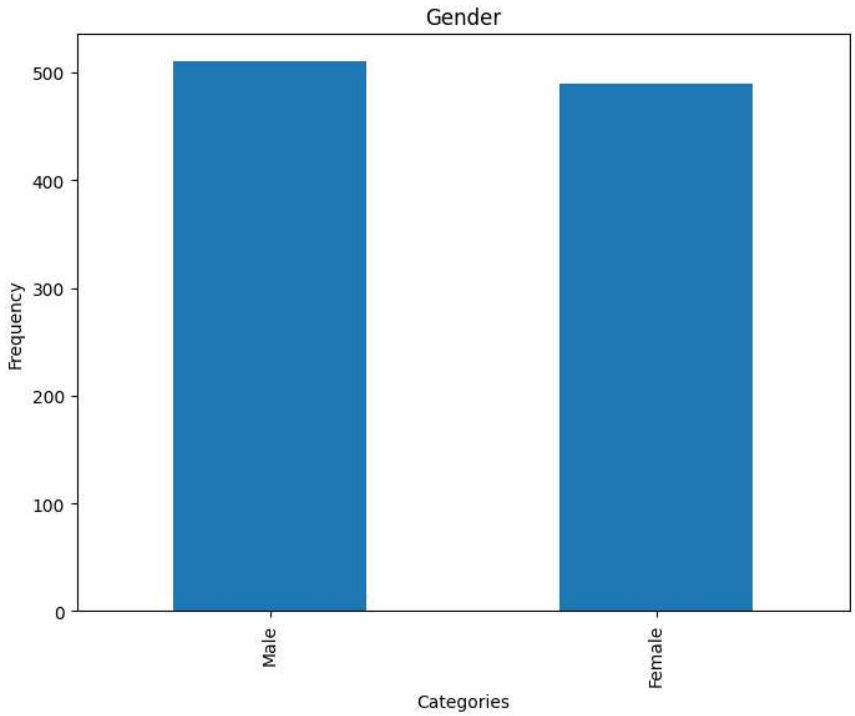
```
import matplotlib.pyplot as plt
import seaborn as sns

# Initial Visualization
# Histograms for numerical variables
df.hist(figsize=(10, 8))
plt.tight_layout()
plt.show()

# Bar plots for categorical variables
categorical_cols = df.select_dtypes(include=['object']).columns
for col in categorical_cols:
    df[col].value_counts().plot(kind='bar', figsize=(8, 6))
    plt.title(col)
    plt.xlabel('Categories')
    plt.ylabel('Frequency')
    plt.show()

# Box plots for numerical variables
df.boxplot(figsize=(10, 6))
plt.show()

# Exploratory Data Analysis (EDA)
# Scatter plots for pairwise relationships
sns.pairplot(df)
plt.show()

# Correlation matrix
corr_matrix = df.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

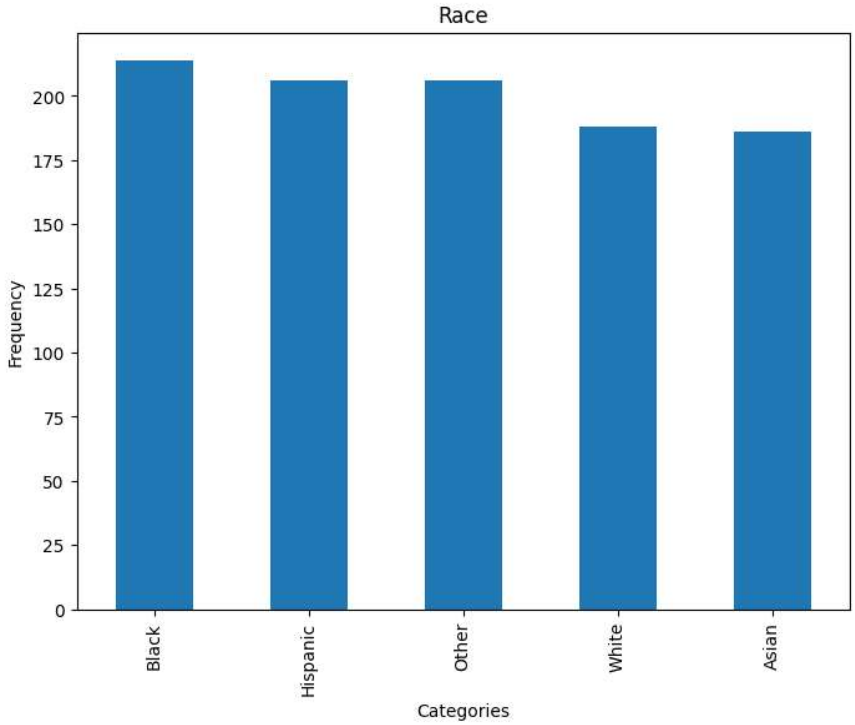

Age
Temperature
Blood Pressure (Systolic)
Blood Pressure (Diastolic)
Height
Weight
Needs Medical Attention

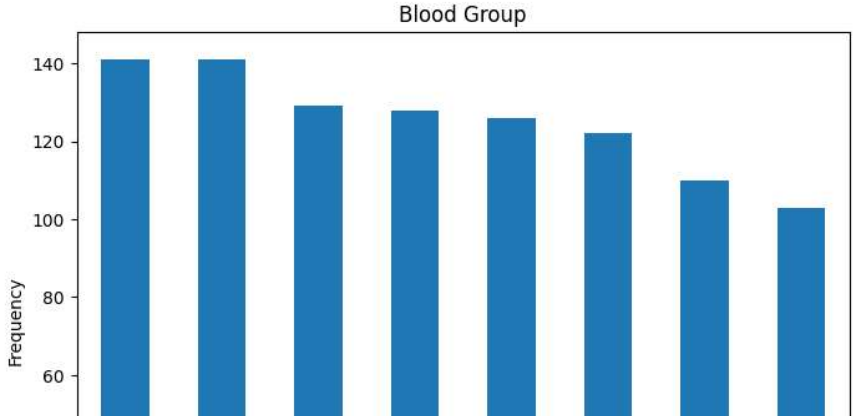

Name
Frequency
Categories

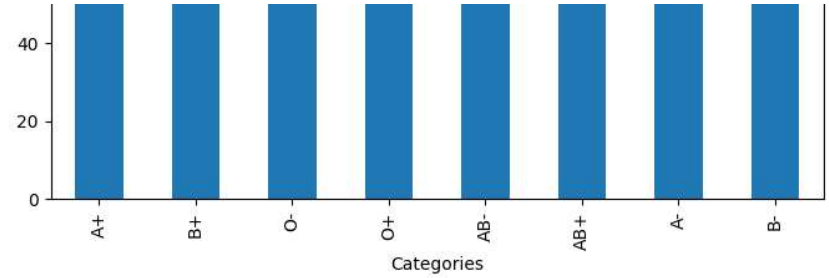

Age Group
Frequency

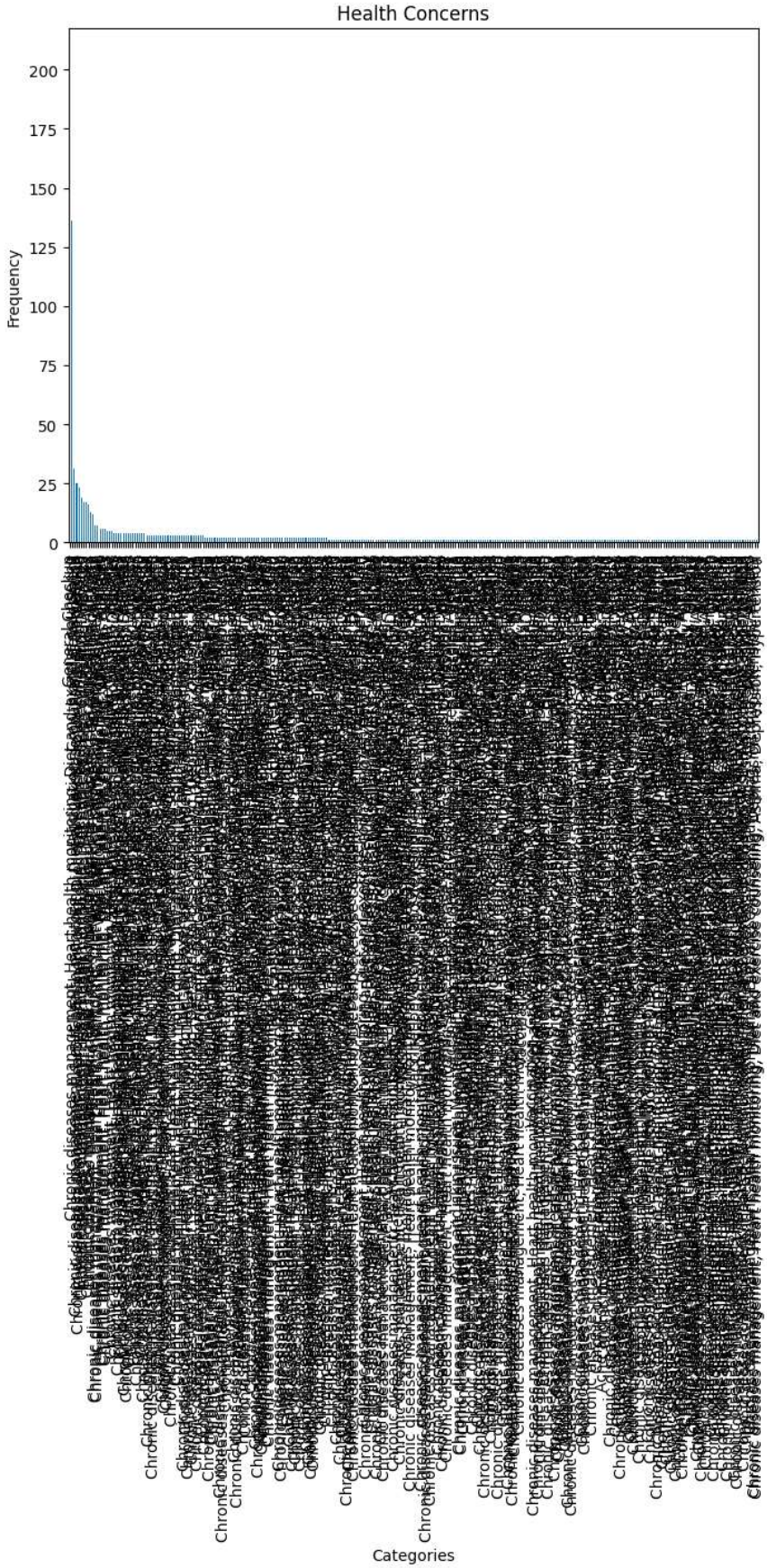

100
0
Adult
Child
Teenager
Unknown
Categories

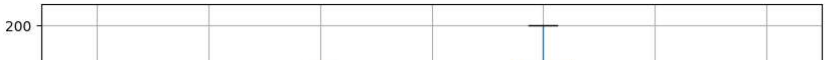

Gender
500
400
300
200
100
0
Frequency
Male
Female
Categories

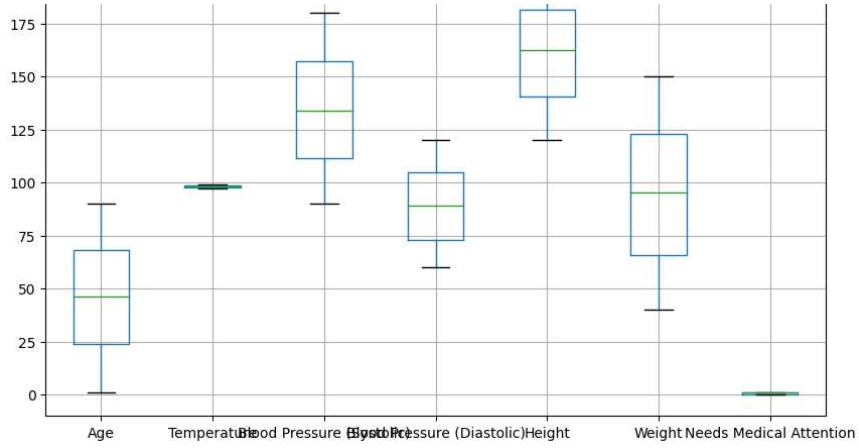

Race
200
175
150
125
100
75
50
25
0
Frequency
Black
Hispanic
Other
White
Asian
Categories

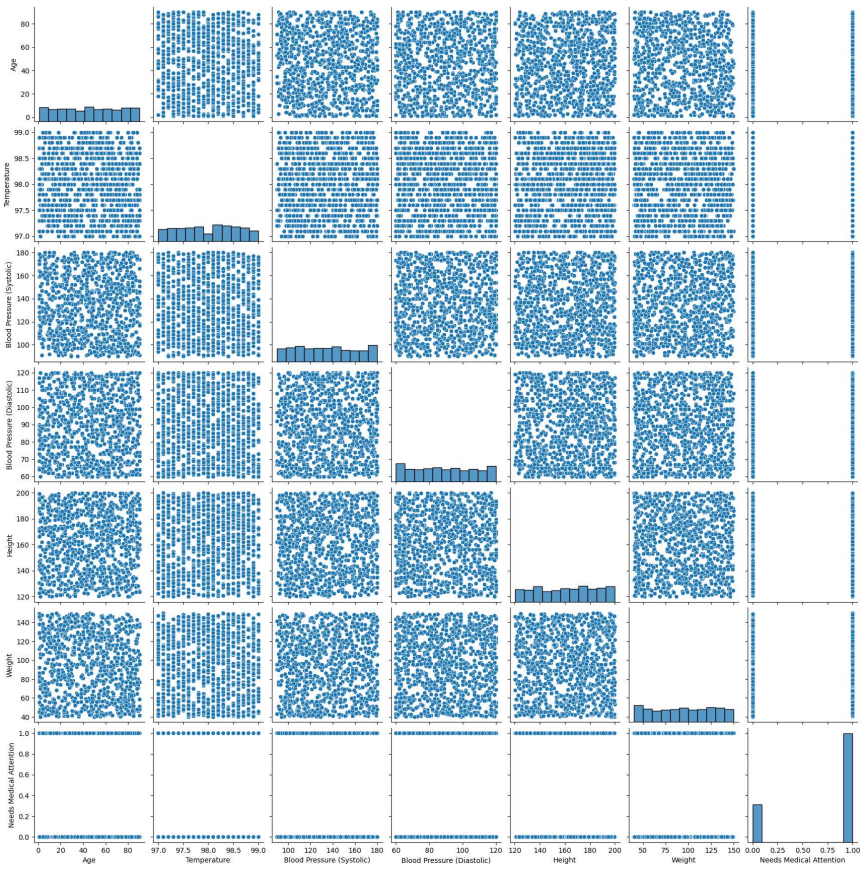

Blood Group
140
120
100
80
60
Frequency


40
20
0
A+
B+
O-
O+
AB-
AB+
A-
B-
Categories

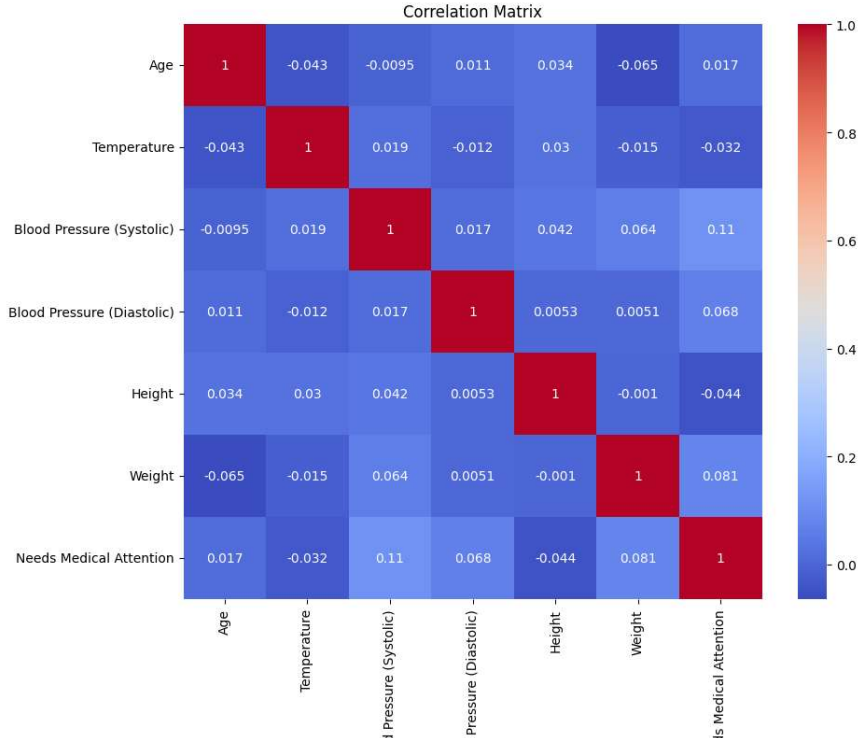

Health Concerns
Frequency
200
175
150
125
100
75
50
25
0
Categories


200






```
<ipython-input-54-6c5d4b929b0f>:29: FutureWarning: The default value of numeric_only
  corr_matrix = df.corr()
```

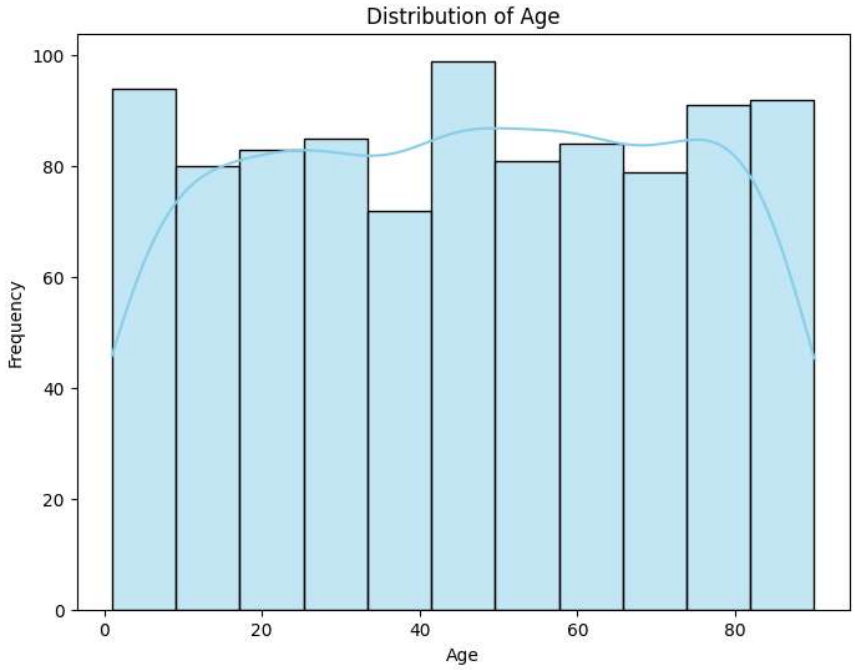


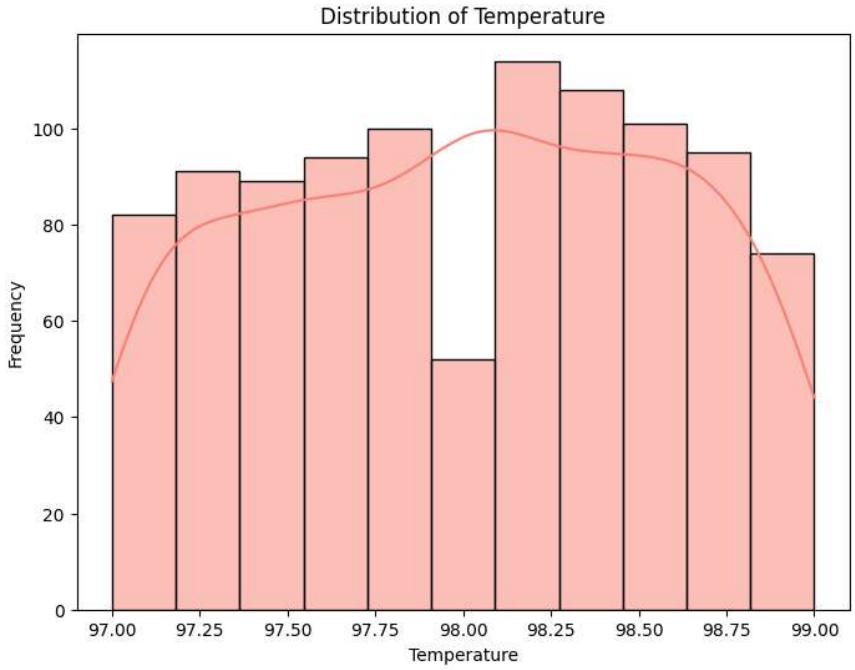

```
import pandas as pd

# Selecting numerical columns for skewness calculation
numerical_columns = df.select_dtypes(include=['int64', 'float64'])
```

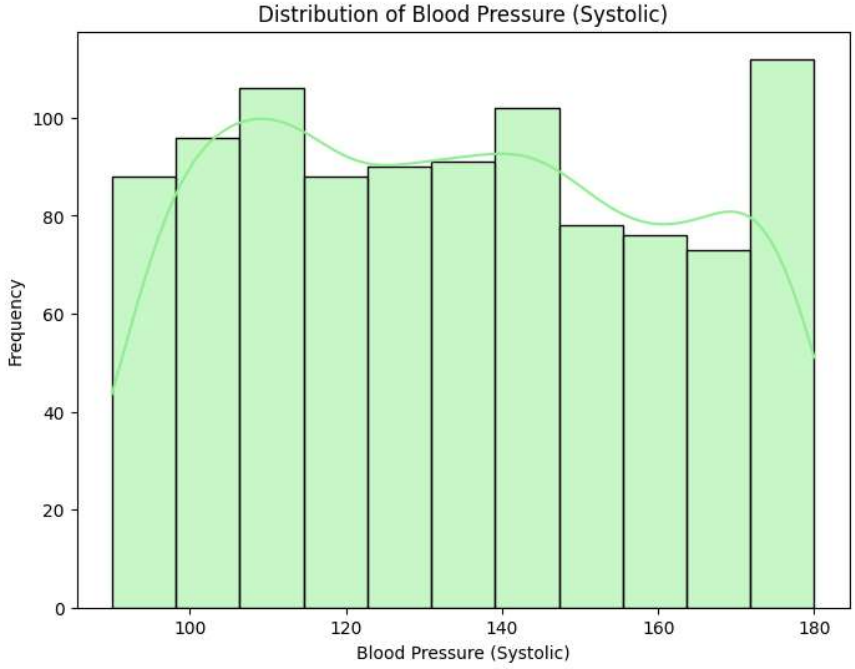
```
# Calculating skewness for each numerical column
skewness = numerical_columns.skew()

# Displaying skewness for each column
print("Skewness for numerical columns:")
print(skewness)
```

```
Skewness for numerical columns:
Age                          -0.029920
Temperature                  -0.060163
Blood Pressure (Systolic)     0.092493
Blood Pressure (Diastolic)    0.054590
Height                       -0.068798
Weight                       -0.048330
Needs Medical Attention      -1.113852
dtype: float64
```

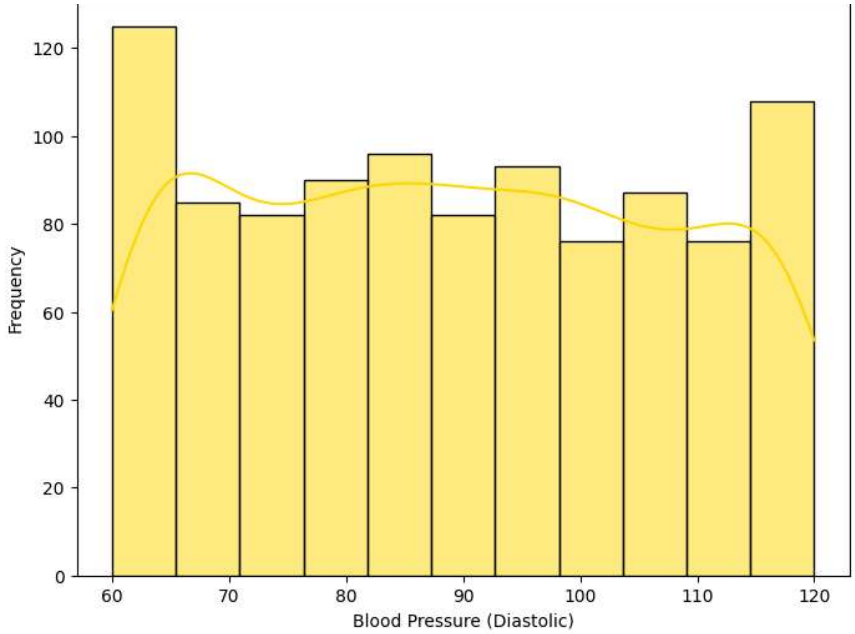
```
import matplotlib.pyplot as plt
import seaborn as sns

# Select numerical columns
numerical_columns = ['Age', 'Temperature', 'Blood Pressure (Systolic)', 'Blood Pressure (Diastolic)', 'Height', 'Weight', 'Needs Medica

# Define bright colors
bright_colors = ['skyblue', 'salmon', 'lightgreen', 'gold', 'lightcoral', 'lightblue', 'lightsalmon']

# Plot histograms for numerical columns with bright colors
for i, column in enumerate(numerical_columns):
    plt.figure(figsize=(8, 6))
    sns.histplot(df[column], kde=True, color=bright_colors[i])
    plt.title(f'Distribution of {column}')
    plt.xlabel(column)
    plt.ylabel('Frequency')
    plt.show()
```

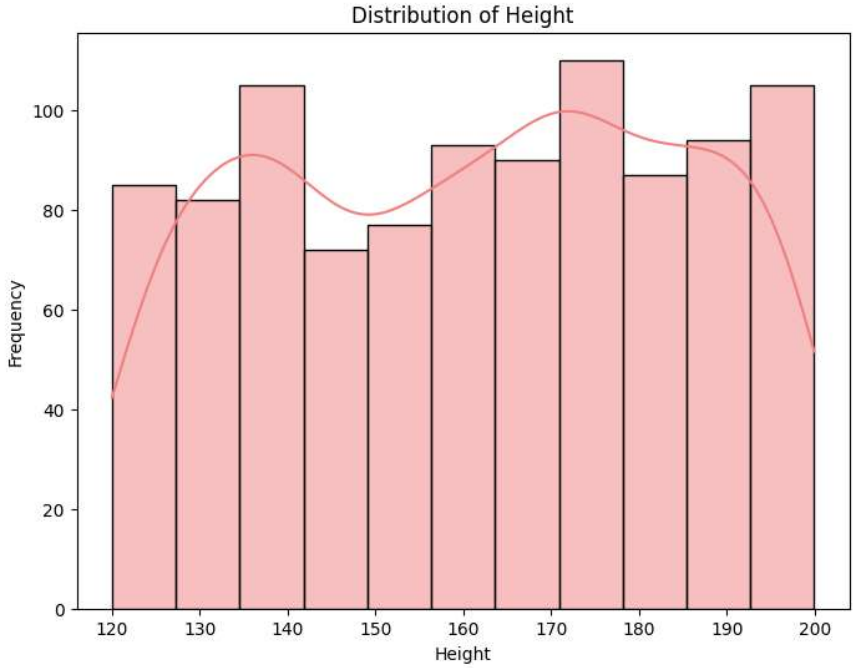


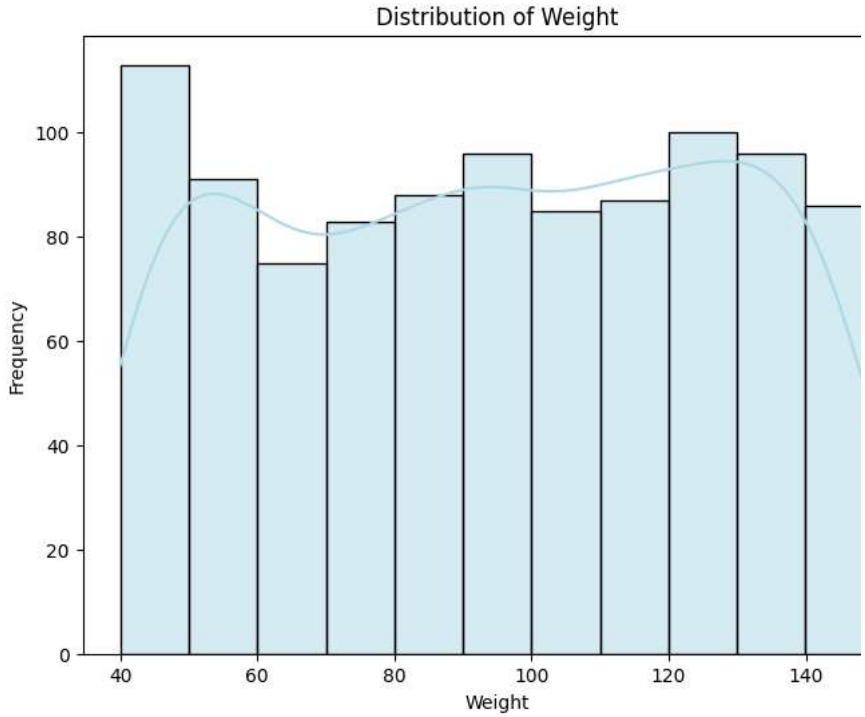




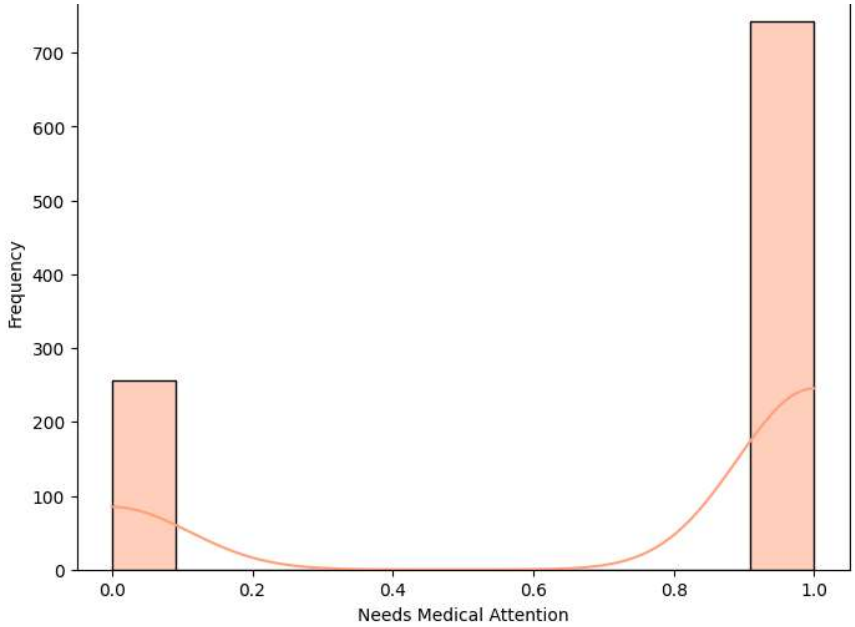
Distribution of Blood Pressure (Diastolic)







Distribution of Needs Medical Attention




```
# Select rows where Age column has NaN values
missing_age_rows = df[df['Age'].isnull()]

# Display rows with missing Age values
print("Rows with missing Age values:")
print(missing_age_rows)
```

```
729                     1
731                     0
744                     1
765                     1
783                     1
807                     1
808                     1
822                     1
833                     0
847                     1
852                     0
862                     1
895                     0
951                     1
956                     1
959                     1
965                     1
992                     0
```

```
# Import necessary libraries
import pandas as pd

# Calculate the mean of the Age column
age_mean = df['Age'].mean()

# Impute missing values in the Age column with the mean
df['Age'].fillna(age_mean, inplace=True)
```

```
# Displaying only the 'Age' column
print(df['Age'])
```

```
0      29.0
1      84.0
2      26.0
3      42.0
4      24.0
       ...
995    36.0
996    15.0
997    21.0
998    89.0
999     1.0
Name: Age, Length: 1000, dtype: float64
```

```
# Define a dictionary mapping age ranges to numeric categories
age_category_dict_numeric = {
    (0, 25): 0,
    (26, 40): 1,
    (41, 60): 2,
    (61, 100): 3
}

# Function to map age to numeric category using the dictionary
def categorize_age_numeric(age):
    for age_range, category in age_category_dict_numeric.items():
        if age_range[0] <= age <= age_range[1]:
            return category
    return -1  # Handle unknown values

# Apply the function to create a new column 'Numeric Age Group'
df['Numeric Age Group'] = df['Age'].apply(categorize_age_numeric)
```

```
df
```

	Name	Age	Age Group	Gender	Race	Blood Group	Temperature	Blood Pressure (Systolic)	E Pres (Diasto
0	Kevin Wong	29.0	Adult	Female	Black	AB+	97.9	178	
1	Mr. David Meyers	84.0	Adult	Male	White	A-	98.1	130	
2	Robert Sanchez DDS	26.0	Adult	Male	Hispanic	A+	97.9	121	
3	Brent Moore	42.0	Adult	Male	Black	B+	98.1	169	
4	Alicia Kennedy	24.0	Adult	Male	Black	A+	98.0	100	
...	...	...	...	...	...	...	...	...	
995	Stacy Gallegos	36.0	Adult	Female	Black	A+	97.9	159	
996	Natalie Curtis	15.0	Teenager	Female	Asian	AB+	98.0	97	
997	Taylor Meyer	21.0	Adult	Male	Hispanic	A+	97.6	119	
998	Erin Morales	89.0	Adult	Male	Hispanic	A+	98.3	93	
999	Gary Harper	1.0	Child	Male	Asian	B+	98.0	128	

1000 rows × 14 columns

```
# Create a dictionary to map gender categories to numeric values
gender_mapping = {'Male': 0, 'Female': 1}

# Map the 'Gender' column using the dictionary
df['Gender'] = df['Gender'].map(gender_mapping)
df
```

	Name	Age	Age Group	Gender	Race	Blood Group	Temperature	Blood Pressure (Systolic)	B Pres (Diastc
0	Kevin Wong	29.0	Adult	1	Black	AB+	97.9	178	
1	Mr. David Meyers	84.0	Adult	0	White	A-	98.1	130	
2	Robert Sanchez DDS	26.0	Adult	0	Hispanic	A+	97.9	121	
3	Brent Moore	42.0	Adult	0	Black	B+	98.1	169	
4	Alicia Kennedy	24.0	Adult	0	Black	A+	98.0	100	
...	...	...	...	...	...	...	...	...	
995	Stacy Gallegos	36.0	Adult	1	Black	A+	97.9	159	
996	Natalie Curtis	15.0	Teenager	1	Asian	AB+	98.0	97	
997	Taylor Meyer	21.0	Adult	0	Hispanic	A+	97.6	119	
998	Erin Morales	89.0	Adult	0	Hispanic	A+	98.3	93	
999	Gary Harper	1.0	Child	0	Asian	B+	98.0	128	

1000 rows × 14 columns

```
# Get unique values in the 'Race' column
unique_races = df['Race'].unique()
print(unique_races)
```

```
['Black' 'White' 'Hispanic' 'Asian' 'Other']
```

```
# Define a dictionary to map race values to numerical categories
race_mapping = {'Black': 0, 'Hispanic': 1, 'Asian': 2, 'Other': 3, 'White': 4}

# Map the 'Race' column using the defined dictionary
df['Race'] = df['Race'].map(race_mapping)

# Display the updated DataFrame
print(df.head())
```

```
                Name   Age Age Group  Gender  Race Blood Group  Temperature  \
0         Kevin Wong  29.0     Adult       1     0         AB+         97.9
1    Mr. David Meyers  84.0     Adult       0     4          A-         98.1
2  Robert Sanchez DDS  26.0     Adult       0     1          A+         97.9
3        Brent Moore  42.0     Adult       0     0          B+         98.1
4     Alicia Kennedy  24.0     Adult       0     0          A+         98.0

   Blood Pressure (Systolic)  Blood Pressure (Diastolic)  Height  Weight  \
0                        178                          84  183.19   73.15
1                        130                          88  170.96   92.36
2                        121                         101  146.33   77.41
3                        169                         112  197.98  113.40
4                        100                          61  149.24   79.90
```

```
                                    Health Concerns  Needs Medical Attention  \
0  Chronic diseases management, Heart health moni...                        1
1  Chronic diseases management, Heart health moni...                        1
2  Chronic diseases management, Heart health moni...                        1
3  Chronic diseases management, Heart health moni...                        1
4  Chronic diseases management, Heart health moni...                        1

   Numeric Age Group
0                  1
1                  3
2                  1
3                  2
4                  0
```

```
# Find unique values in the 'Blood Group' column
unique_blood_groups = df['Blood Group'].unique()

# Display the unique values
print(unique_blood_groups)
```

```
['AB+' 'A-' 'A+' 'B+' 'B-' 'AB-' 'O+' 'O-']
```

```
# Define the mapping dictionary for blood group
blood_group_mapping = {'AB-': 0, 'A-': 1, 'O+': 2, 'B+': 3, 'A+': 4, 'O-': 5, 'AB+': 6, 'B-': 7}

# Map the values in the "Blood Group" column to numeric values
df['Blood Group'] = df['Blood Group'].map(blood_group_mapping)
df
```

	Name	Age	Age Group	Gender	Race	Blood Group	Temperature	Blood Pressure (Systolic)	Bloc Pressur (Diastolic
0	Kevin Wong	29.0	Adult	1	0	6	97.9	178	8
1	Mr. David Meyers	84.0	Adult	0	4	1	98.1	130	8
2	Robert Sanchez DDS	26.0	Adult	0	1	4	97.9	121	1(
3	Brent Moore	42.0	Adult	0	0	3	98.1	169	11
4	Alicia Kennedy	24.0	Adult	0	0	4	98.0	100	6
...	...	...	...	...	...	...	...	...	
995	Stacy Gallegos	36.0	Adult	1	0	4	97.9	159	6
996	Natalie Curtis	15.0	Teenager	1	2	6	98.0	97	6
997	Taylor Meyer	21.0	Adult	0	1	4	97.6	119	11
998	Erin Morales	89.0	Adult	0	1	4	98.3	93	8
999	Gary Harper	1.0	Child	0	2	3	98.0	128	8

1000 rows × 14 columns

```
import pandas as pd

# Fill blank rows with 'Unknown'
df['Health Concerns'].fillna('Unknown', inplace=True)

# Convert the 'Health Concerns' column to a list of health concerns
df['Health Concerns'] = df['Health Concerns'].str.split(',')

# Display the first few rows to verify the conversion
print(df.head())

df
```

```
                Name   Age Age Group  Gender  Race  Blood Group  Temperature  \
0         Kevin Wong  29.0     Adult       1     0            6         97.9
1    Mr. David Meyers  84.0     Adult       0     4            1         98.1
2  Robert Sanchez DDS  26.0     Adult       0     1            4         97.9
3        Brent Moore  42.0     Adult       0     0            3         98.1
4     Alicia Kennedy  24.0     Adult       0     0            4         98.0

   Blood Pressure (Systolic)  Blood Pressure (Diastolic)  Height  Weight  \
0                        178                          84  183.19   73.15
1                        130                          88  170.96   92.36
2                        121                         101  146.33   77.41
3                        169                         112  197.98  113.40
4                        100                          61  149.24   79.90

                                 Health Concerns  Needs Medical Attention  \
0  [Chronic diseases management,  Heart health mo...                       1
1  [Chronic diseases management,  Heart health mo...                       1
2  [Chronic diseases management,  Heart health mo...                       1
3  [Chronic diseases management,  Heart health mo...                       1
4  [Chronic diseases management,  Heart health mo...                       1

   Numeric Age Group
0                  1
1                  3
2                  1
3                  2
4                  0
```

	Name	Age	Age Group	Gender	Race	Blood Group	Temperature	Blood Pressure (Systolic)	Bloc Pressur (Diastolic
0	Kevin Wong	29.0	Adult	1	0	6	97.9	178	8
1	Mr. David Meyers	84.0	Adult	0	4	1	98.1	130	8
2	Robert Sanchez DDS	26.0	Adult	0	1	4	97.9	121	10
3	Brent Moore	42.0	Adult	0	0	3	98.1	169	11
4	Alicia Kennedy	24.0	Adult	0	0	4	98.0	100	6
...	...	...	...	...	...	...	...	...	
995	Stacy Gallegos	36.0	Adult	1	0	4	97.9	159	6
996	Natalie Curtis	15.0	Teenager	1	2	6	98.0	97	6
997	Taylor Meyer	21.0	Adult	0	1	4	97.6	119	11
998	Erin Morales	89.0	Adult	0	1	4	98.3	93	8
999	Gary Harper	1.0	Child	0	2	3	98.0	128	8

1000 rows × 14 columns

```
df['Health Concerns']
```

```
0      [Chronic diseases management,  Heart health mo...
1      [Chronic diseases management,  Heart health mo...
2      [Chronic diseases management,  Heart health mo...
```

```
3      [Chronic diseases management,  Heart health mo...
4      [Chronic diseases management,  Heart health mo...
                              ...
995    [Chronic diseases management,  Heart health mo...
996    [Acne and skin issues,  Mental health challeng...
997                                    [General Checkup]
998                                    [General Checkup]
999                                    [General Checkup]
Name: Health Concerns, Length: 1000, dtype: object
```

```
import pandas as pd
# Fill blank cells with "Unknown"
df['Health Concerns'].fillna('Unknown', inplace=True)

# Split rows with multiple health conditions by commas and create a new column
df['Health Concern'] = df['Health Concerns'].str.split(',')

# Iterate over each row in the original DataFrame and handle missing values
new_rows = []

for index, row in df.iterrows():
    health_concerns = row['Health Concern']
    if isinstance(health_concerns, list):
        for health_condition in health_concerns:
            new_row = row.copy()
            new_row['Health Concern'] = health_condition.strip() if isinstance(health_condition, str) else 'Unknown'
            new_rows.append(new_row)
    else:
        # If there's no multiple health condition, keep the row with 'Health Concern' as is
        new_row = row.copy()
        new_row['Health Concern'] = health_concerns.strip() if isinstance(health_concerns, str) else 'Unknown'
        new_rows.append(new_row)

# Create a new DataFrame from the list of duplicated rows
new_df = pd.DataFrame(new_rows)

# Counting the number of health concerns in each row
new_df['Number of Health Concerns'] = new_df['Health Concerns'].apply(len)

# Exploring unique health concerns
unique_health_concerns = set()
for health_concerns in new_df['Health Concerns']:
    unique_health_concerns.update(health_concerns)

# Print unique health concerns
print("Unique Health Concerns:", unique_health_concerns)

# Creating a numerical representation using one-hot encoding
for concern in unique_health_concerns:
    new_df[concern] = new_df['Health Concerns'].apply(lambda x: 1 if concern in x else 0)
```

```
Unique Health Concerns: {'Asthma', 'Allergies', ' Depression', 'Diabetes', ' Sexual health education', ' Heart health monitoring',
```

```
df
```

	Name	Age	Age Group	Gender	Race	Blood Group	Temperature	Blood Pressure (Systolic)	Bloc Pressur (Diastolic
0	Kevin Wong	29.0	Adult	1	0	6	97.9	178	8
1	Mr. David Meyers	84.0	Adult	0	4	1	98.1	130	8
2	Robert Sanchez DDS	26.0	Adult	0	1	4	97.9	121	10
3	Brent Moore	42.0	Adult	0	0	3	98.1	169	11
4	Alicia Kennedy	24.0	Adult	0	0	4	98.0	100	6
...	...	...	...	...	...	...	...	...	
995	Stacy Gallegos	36.0	Adult	1	0	4	97.9	159	6
996	Natalie Curtis	15.0	Teenager	1	2	6	98.0	97	6
997	Taylor Meyer	21.0	Adult	0	1	4	97.6	119	11
998	Erin Morales	89.0	Adult	0	1	4	98.3	93	8
999	Gary Harper	1.0	Child	0	2	3	98.0	128	8

1000 rows × 15 columns

```
df
```

	Name	Age	Age Group	Gender	Race	Blood Group	Temperature	Blood Pressure (Systolic)	Bloc Pressur (Diastolic
0	Kevin Wong	29.0	Adult	1	0	6	97.9	178	8
1	Mr. David Meyers	84.0	Adult	0	4	1	98.1	130	8
2	Robert Sanchez DDS	26.0	Adult	0	1	4	97.9	121	10
3	Brent Moore	42.0	Adult	0	0	3	98.1	169	11
4	Alicia Kennedy	24.0	Adult	0	0	4	98.0	100	6
...	...	...	...	...	...	...	...	...	
995	Stacy Gallegos	36.0	Adult	1	0	4	97.9	159	6
996	Natalie Curtis	15.0	Teenager	1	2	6	98.0	97	6
997	Taylor Meyer	21.0	Adult	0	1	4	97.6	119	11
998	Erin Morales	89.0	Adult	0	1	4	98.3	93	8
999	Gary Harper	1.0	Child	0	2	3	98.0	128	8

1000 rows × 15 columns

```
# Counting the number of health concerns in each row
df['Number of Health Concerns'] = df['Health Concerns'].apply(len)

# Exploring unique health concerns
unique_health_concerns = set()
for health_concerns in df['Health Concerns']:
    unique_health_concerns.update(health_concerns)

# Print unique health concerns
print("Unique Health Concerns:", unique_health_concerns)

# Creating a numerical representation using one-hot encoding
for concern in unique_health_concerns:
    df[concern] = df['Health Concerns'].apply(lambda x: 1 if concern in x else 0)
```

```
Unique Health Concerns: {'Asthma', 'Allergies', ' Depression', 'Diabetes', ' Sexual health education', ' Heart health monitoring',
```

```
df ['Health Concern']
```

```
0     NaN
1     NaN
2     NaN
3     NaN
4     NaN
      ..
995   NaN
996   NaN
```

```
997   NaN
998   NaN
999   NaN
Name: Health Concern, Length: 1000, dtype: float64
```

```
import pandas as pd
# Drop the 'Health Concern' column
df.drop(columns=['Health Concern'], inplace=True)
```

```
column_names = df.columns
print(column_names)
```

```
Index(['Name', 'Age', 'Age Group', 'Gender', 'Race', 'Blood Group',
       'Temperature', 'Blood Pressure (Systolic)',
       'Blood Pressure (Diastolic)', 'Height', 'Weight', 'Health Concerns',
       'Needs Medical Attention', 'Numeric Age Group',
       'Number of Health Concerns', 'Asthma', 'Allergies', ' Depression',
       'Diabetes', ' Sexual health education', ' Heart health monitoring',
       'Chronic diseases management', 'Anxiety', ' Diabetes', 'Obesity',
       'Depression', ' Mental health challenges', ' Growth and development',
       ' Obesity', ' Anxiety', ' Hypertension', ' Allergies', 'Vaccination',
       'Unknown', ' Diet and exercise counseling', 'Acne and skin issues',
       'General Checkup', 'Hypertension', ' Asthma'],
      dtype='object')
```

```
# Select only numeric columns from the DataFrame
numeric_df = df.select_dtypes(include='number')

# Display the selected numeric columns
print(numeric_df)
numeric_df.corr()
```

```
      Age  Gender  Race  Blood Group  Temperature  Blood Pressure (Systolic)  \
0    29.0       1     0            6         97.9                        178
1    84.0       0     4            1         98.1                        130
2    26.0       0     1            4         97.9                        121
3    42.0       0     0            3         98.1                        169
4    24.0       0     0            4         98.0                        100
..    ...     ...   ...          ...          ...                        ...
995  36.0       1     0            4         97.9                        159
996  15.0       1     2            6         98.0                         97
997  21.0       0     1            4         97.6                        119
998  89.0       0     1            4         98.3                         93
999   1.0       0     2            3         98.0                        128

     Blood Pressure (Diastolic)  Height  Weight  Needs Medical Attention  ...  \
0                            84  183.19   73.15                        1  ...
1                            88  170.96   92.36                        1  ...
2                           101  146.33   77.41                        1  ...
3                           112  197.98  113.40                        1  ...
4                            61  149.24   79.90                        1  ...
..                          ...     ...     ...                      ...  ...
995                          68  162.34  136.16                        1  ...
996                          69  182.38   55.07                        0  ...
997                         112  185.71  117.14                        0  ...
998                          84  144.21  109.79                        0  ...
999                          81  194.56   77.49                        0  ...

      Anxiety   Hypertension   Allergies  Vaccination  Unknown  \
0           0              0           0            0        0
1           0              0           0            0        0
2           1              0           0            0        0
3           0              0           0            0        0
4           0              0           0            0        0
..        ...            ...         ...          ...      ...
995         0              0           0            0        0
996         0              0           0            0        0
997         0              0           0            0        0
998         0              0           0            0        0
999         0              0           0            0        0

      Diet and exercise counseling  Acne and skin issues  General Checkup  \
0                                1                     0                0
1                                1                     0                0
2                                1                     0                0
3                                1                     0                0
4                                1                     0                0
..                             ...                   ...              ...
995                              1                     0                0
996                              0                     1                0
997                              0                     0                1
998                              0                     0                1
999                              0                     0                1

     Hypertension   Asthma
0               0        0
1               0        1
2               0        0
3               0        0
4               0        0
..            ...      ...
995             0        0
996             0        0
997             0        0
998             0        0
999             0        0

[1000 rows x 36 columns]
```

	Age	Gender	Race	Blood Group	Temperature	Blood Pressure (Systolic)	(D
Age	1.000000e+00	0.021437	-0.015645	-0.042293	-0.041872	-0.009177	
Gender	2.143728e-02	1.000000	0.018985	0.052414	-0.011855	-0.048921	
Race	-1.564549e-02	0.018985	1.000000	0.037033	-0.006760	0.057787	
Blood Group	-4.229306e-02	0.052414	0.037033	1.000000	0.021326	-0.012643	
Temperature	-4.187165e-02	-0.011855	-0.006760	0.021326	1.000000	0.019296	
Blood Pressure (Systolic)	-9.177439e-03	-0.048921	0.057787	-0.012643	0.019296	1.000000	
Blood Pressure (Diastolic)	1.062430e-02	0.012449	-0.016144	-0.029204	-0.012202	0.016760	

Height	3.325761e-02	0.047283	-0.008441	0.028555	0.029523	0.042426
Weight	-6.301085e-02	0.022803	-0.003821	-0.040494	-0.015252	0.063763
Needs Medical Attention	1.622124e-02	0.008835	0.020359	-0.006298	-0.031977	0.110760
Numeric Age Group	9.564945e-01	0.014556	-0.010882	-0.054028	-0.036830	0.013946
Number of Health Concerns	1.366514e-01	0.012977	0.007749	0.042663	-0.019376	0.044720
Asthma	-4.233181e-17	-0.008684	0.063130	-0.038544	0.033460	-0.020945
Allergies	1.271284e-17	-0.062117	0.002320	-0.020065	-0.006524	-0.038536
Depression	3.880559e-03	0.033578	0.019373	0.004919	-0.031516	-0.014142
Diabetes	3.544219e-17	0.000896	-0.014120	-0.009119	0.045991	-0.047740
Sexual health education	-2.616739e-01	-0.056995	-0.015157	0.050800	-0.003553	-0.049230
Heart health monitoring	4.541202e-01	0.054440	-0.016273	0.018692	-0.019152	0.011802
Chronic diseases management	4.541202e-01	0.054440	-0.016273	0.018692	-0.019152	0.011802
Anxiety	-5.910761e-17	0.015599	-0.037333	-0.052863	-0.030719	0.005900
Diabetes	3.286694e-02	-0.005674	0.021251	0.021311	-0.013531	0.005607
Obesity	2.707932e-18	0.015599	0.052505	-0.008037	-0.006065	0.005900
Depression	2.219259e-17	-0.012762	-0.017369	0.023982	0.030916	-0.059081
Mental health challenges	-2.616739e-01	-0.056995	-0.015157	0.050800	-0.003553	-0.049230
Growth and development	-5.342369e-01	-0.012844	0.042457	-0.025012	0.021323	0.023675
Obesity	7.392585e-02	0.014150	-0.011165	-0.006511	0.035502	0.009684
Anxiety	-3.643369e-02	0.030603	-0.003444	0.000919	0.004035	0.032617
Hypertension	8.153629e-02	0.005562	0.007411	0.001016	-0.008159	0.019658
Allergies	3.449431e-02	-0.034372	-0.024915	0.002059	0.015142	0.046206
Vaccination	-5.342369e-01	-0.012844	0.042457	-0.025012	0.021323	0.023675
Unknown	1.959636e-17	-0.013392	-0.034610	0.013621	-0.046388	0.087443
Diet and exercise counseling	4.541202e-01	0.054440	-0.016273	0.018692	-0.019152	0.011802
Acne and skin issues	-2.616739e-01	-0.056995	-0.015157	0.050800	-0.003553	-0.049230
General Checkup	-1.227239e-02	-0.011998	-0.002148	-0.012661	0.015432	-0.020565
Hypertension	-1.316623e-18	-0.010317	-0.022261	-0.011686	-0.013858	0.055456
Asthma	-1.285699e-02	-0.023955	0.041015	0.033228	-0.054645	0.028242

36 rows × 36 columns

```
column_names1 = numeric_df.columns
print(column_names1)
```

```
Index(['Age', 'Gender', 'Race', 'Blood Group', 'Temperature',
       'Blood Pressure (Systolic)', 'Blood Pressure (Diastolic)', 'Height',
       'Weight', 'Needs Medical Attention', 'Numeric Age Group',
       'Number of Health Concerns', ' Mental health challenges',
       ' Growth and development', 'Vaccination', ' Obesity', ' Diabetes',
       ' Depression', ' Anxiety', ' Hypertension',
       ' Diet and exercise counseling', ' Allergies',
       ' Sexual health education', 'Acne and skin issues',
       'Chronic diseases management', ' Heart health monitoring', ' Asthma'],
      dtype='object')
```

```
# Separate the input variables (features) from the target variable
X = numeric_df.drop(columns=['Needs Medical Attention'])  # Input variables
y = numeric_df['Needs Medical Attention']  # Target variable

# Display the input variables and target variable
print("Input variables (features):")
print(X)
print("\nTarget variable:")
print(y)
```

```
997                        112  185.71  117.14                 0  ...
998                         84  144.21  109.79                 3  ...
999                         81  194.56   77.49                 0  ...

      Anxiety   Hypertension   Allergies  Vaccination  Unknown  \
0           0              0           0            0        0
1           0              0           0            0        0
2           1              0           0            0        0
3           0              0           0            0        0
4           0              0           0            0        0
..        ...            ...         ...          ...      ...
995         0              0           0            0        0
996         0              0           0            0        0
997         0              0           0            0        0
998         0              0           0            0        0
999         0              0           0            0        0

      Diet and exercise counseling  Acne and skin issues  General Checkup  \
0                                1                     0                0
1                                1                     0                0
2                                1                     0                0
3                                1                     0                0
4                                1                     0                0
..                             ...                   ...              ...
995                              1                     0                0
996                              0                     1                0
997                              0                     0                1
998                              0                     0                1
999                              0                     0                1

     Hypertension   Asthma
0               0        0
1               0        1
2               0        0
3               0        0
4               0        0
..            ...      ...
995             0        0
996             0        0
997             0        0
998             0        0
999             0        0

[1000 rows x 35 columns]

Target variable:
0      1
1      1
2      1
3      1
4      1
      ..
995    1
996    0
997    0
998    0
999    0
Name: Needs Medical Attention, Length: 1000, dtype: int64
```

```
from sklearn.model_selection import train_test_split
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
# Display the shapes of the training and testing sets
print("Training set - X shape:", X_train.shape)
```

```
print("Training set - y shape:", y_train.shape)
print("Testing set - X shape:", X_test.shape)
print("Testing set - y shape:", y_test.shape)
```

```
Training set - X shape: (800, 35)
Training set - y shape: (800,)
Testing set - X shape: (200, 35)
Testing set - y shape: (200,)
```

```
from sklearn.linear_model import LogisticRegression
# Create the model
logistic_regression_model = LogisticRegression(solver='saga', max_iter=1000)

# Train the model
logistic_regression_model.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_sag.py:350: Convergenc
  warnings.warn(
▾ LogisticRegression
LogisticRegression(max_iter=1000, solver='saga')
```

```
from sklearn.tree import DecisionTreeClassifier

# Create the model
decision_tree_model = DecisionTreeClassifier()

# Train the model
decision_tree_model.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
from sklearn.ensemble import RandomForestClassifier

# Create the model
random_forest_model = RandomForestClassifier()

# Train the model
random_forest_model.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
from sklearn.neighbors import KNeighborsClassifier

# Create the model
knn_model = KNeighborsClassifier()

# Train the model
knn_model.fit(X_train, y_train)
```

```
▾ KNeighborsClassifier
KNeighborsClassifier()
```

```
# Import necessary libraries
from sklearn.metrics import accuracy_score, classification_report

# Define a function to make predictions and evaluate the model
def evaluate_model(model, X_test, y_test):
    # Make predictions on the testing data
    y_pred = model.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, y_pred)
    print("Accuracy:", accuracy)

    # Display classification report
    print("\nClassification Report:")
    print(classification_report(y_test, y_pred))

    # Return the predictions
    return y_pred

# Evaluate each trained model
models = {
    "Logistic Regression": logistic_regression_model,
    "Decision Tree": decision_tree_model,
    "Random Forest": random_forest_model,
    "K-Nearest Neighbors": knn_model
}

for model_name, model in models.items():
    print("Model:", model_name)
    print("Predictions and Evaluation on Testing Data:")
    y_pred = evaluate_model(model, X_test, y_test)
    print("----------------------------------------\n")
```

```
              0       0.89      0.89      0.89        46
              1       0.97      0.97      0.97       154

       accuracy                           0.95       200
      macro avg       0.93      0.93      0.93       200
   weighted avg       0.95      0.95      0.95       200

   ----------------------------------------

   Model: Decision Tree
   Predictions and Evaluation on Testing Data:
   Accuracy: 0.965

   Classification Report:
                 precision    recall  f1-score   support

              0       0.88      0.98      0.93        46
              1       0.99      0.96      0.98       154

       accuracy                           0.96       200
      macro avg       0.94      0.97      0.95       200
   weighted avg       0.97      0.96      0.97       200

   ----------------------------------------

   Model: Random Forest
   Predictions and Evaluation on Testing Data:
   Accuracy: 0.965

   Classification Report:
                 precision    recall  f1-score   support

              0       1.00      0.85      0.92        46
              1       0.96      1.00      0.98       154

       accuracy                           0.96       200
      macro avg       0.98      0.92      0.95       200
   weighted avg       0.97      0.96      0.96       200

   ----------------------------------------

   Model: K-Nearest Neighbors
   Predictions and Evaluation on Testing Data:
   Accuracy: 0.69

   Classification Report:
                 precision    recall  f1-score   support
```

```
weighted avg       0.66      0.69      0.68       200

-------------------------------------------
```

```
# Filter patients needing medical attention
df_medical_attention = numeric_df[numeric_df['Needs Medical Attention'] == 1]

# Display the subset of patients needing medical attention
print("Patients needing medical attention:")
print(df_medical_attention)
```

```
991  80.0       1     1           4         97.7                       159
993  58.0       0     4           1         98.3                       155
994  28.0       1     3           5         97.6                       124
995  36.0       1     0           4         97.9                       159

     Blood Pressure (Diastolic)  Height  Weight  Needs Medical Attention  ...  \
0                            84  183.19   73.15                        1  ...
1                            88  170.96   92.36                        1  ...
2                           101  146.33   77.41                        1  ...
3                           112  197.98  113.40                        1  ...
4                            61  149.24   79.90                        1  ...
..                          ...     ...     ...                      ...  ...
990                         105  162.68   52.39                        1  ...
991                          83  132.29  148.18                        1  ...
993                          74  163.94  103.98                        1  ...
994                          62  165.12   83.81                        1  ...
995                          68  162.34  136.16                        1  ...

      Anxiety   Hypertension   Allergies  Vaccination  Unknown  \
0           0              0           0            0        0
1           0              0           0            0        0
2           1              0           0            0        0
3           0              0           0            0        0
4           0              0           0            0        0
..        ...            ...         ...          ...      ...
990         0              0           0            0        0
991         1              0           0            0        0
993         0              0           0            0        0
994         0              1           0            0        0
995         0              0           0            0        0

      Diet and exercise counseling  Acne and skin issues  General Checkup  \
0                                1                     0                0
1                                1                     0                0
2                                1                     0                0
3                                1                     0                0
4                                1                     0                0
..                             ...                   ...              ...
990                              1                     0                0
991                              1                     0                0
993                              1                     0                0
994                              1                     0                0
995                              1                     0                0

     Hypertension   Asthma
0               0        0
1               0        1
2               0        0
3               0        0
4               0        0
..            ...      ...
990             0        0
991             0        0
993             0        1
994             0        1
995             0        0

[743 rows x 36 columns]
```

```
from sklearn.cluster import KMeans

# Specify the number of clusters
num_clusters = 5  #can be adjusted based on the dataset and requirements

# Selecting only numeric columns for clustering
numeric_columns = ['Age', 'Temperature', 'Blood Pressure (Systolic)', 'Blood Pressure (Diastolic)', 'Height', 'Weight']

# Extracting numeric data for clustering
X = df_medical_attention[numeric_columns]

# Initialize the KMeans model
kmeans = KMeans(n_clusters=num_clusters, random_state=0)

# Fit the KMeans model to the data
kmeans.fit(X)

# Get the cluster labels
cluster_labels = kmeans.labels_

# Add the cluster labels to the subset dataframe
df_medical_attention['Cluster'] = cluster_labels

# Print the counts of patients in each cluster
print("Number of patients in each cluster:")
print(df_medical_attention['Cluster'].value_counts())
```

```
Number of patients in each cluster:
2    186
3    175
4    134
0    129
1    119
Name: Cluster, dtype: int64
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning: The default value of `n_init` will change fr
  warnings.warn(
<ipython-input-88-357cc1a7d179>:22: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df_medical_attention['Cluster'] = cluster_labels
```

```
# Analyze the clusters created by KMeans

# Calculate the centroids of each cluster
centroids = kmeans.cluster_centers_

# Assign cluster labels to the original dataframe
df_medical_attention['Cluster'] = kmeans.labels_

# Group the dataframe by cluster and calculate summary statistics for each cluster
cluster_summary = df_medical_attention.groupby('Cluster')[numeric_columns].describe()

# Display the cluster centroids
print("Centroids of Each Cluster:")
print(pd.DataFrame(centroids, columns=numeric_columns))

# Display the summary statistics for each cluster
print("\nSummary Statistics for Each Cluster:")
print(cluster_summary)
```

```
<ipython-input-90-2909b773166d>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  df_medical_attention['Cluster'] = kmeans.labels_
Centroids of Each Cluster:
         Age  Temperature  Blood Pressure (Systolic)  \
0  19.419957    98.004651                 142.116279
1  62.587699    97.939496                 108.941176
2  38.530805    98.023656                 113.301075
3  45.397277    97.998286                 159.445714
4  68.205399    98.009701                 155.731343

   Blood Pressure (Diastolic)      Height      Weight
0                   91.124031  158.920078   65.420000
1                   92.571429  156.928992   65.326723
2                   87.360215  162.565054  123.083226
3                   89.965714  158.791829  127.367486
4                   90.022388  165.422985   75.639179

Summary Statistics for Each Cluster:
```

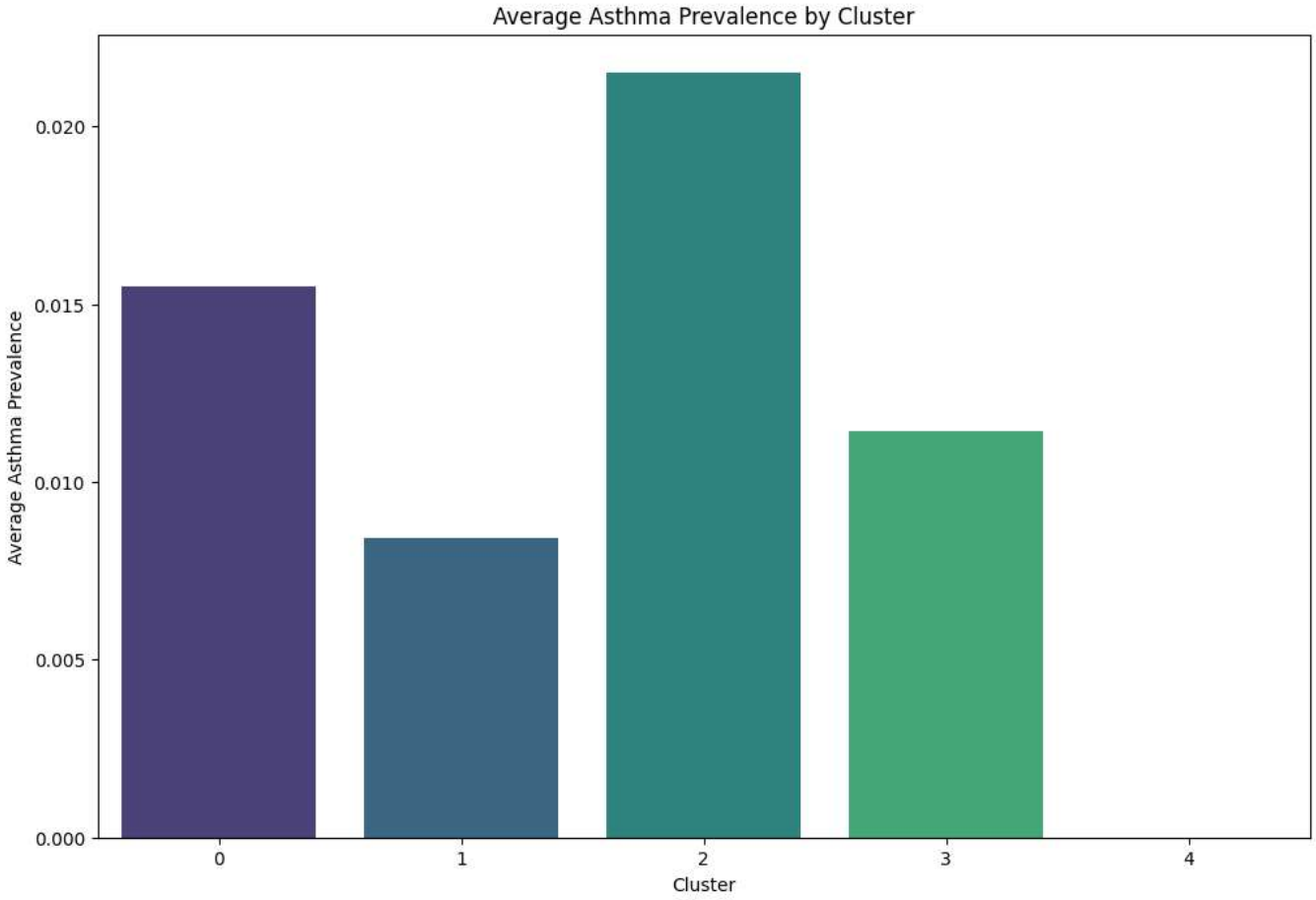
```
                Age                                                    \
              count       mean        std   min    25%   50%   75%   max
Cluster
0         129.0  19.419957  13.604296   1.0   6.00  17.0  30.0  47.0
1         119.0  62.587699  17.466522  24.0  47.00  62.0  77.0  90.0
2         186.0  38.530805  23.925644   1.0  18.00  37.5  56.0  89.0
3         175.0  45.397277  22.407144   1.0  29.00  45.0  61.5  90.0
4         134.0  68.205399  14.912736  39.0  56.25  70.5  81.0  90.0

         Temperature             ...    Height         Weight              \
               count       mean  ...       75%     max  count        mean
Cluster                          ...
0              129.0  98.004651  ...  180.5400  199.48  129.0   65.420000
1              119.0  97.939496  ...  177.3150  199.61  119.0   65.326723
2              186.0  98.023656  ...  181.3825  199.54  186.0  123.083226
3              175.0  97.998286  ...  183.3100  199.67  175.0  127.367486
4              134.0  98.009701  ...  182.9050  199.84  134.0   75.639179


                std    min       25%      50%       75%     max
Cluster
0         17.719571  40.81   49.0100   62.340   80.3300  106.39
1         17.685420  40.26   50.0800   63.030   79.0650  102.96
2         15.287500  84.47  110.4325  124.115  135.5900  149.59
3         14.542984  94.73  116.1650  128.880  140.0600  149.93
4         18.855708  40.17   62.2200   77.000   90.8725  111.36

[5 rows x 48 columns]
```

```
column_labels = df_medical_attention.columns
print(column_labels)
```

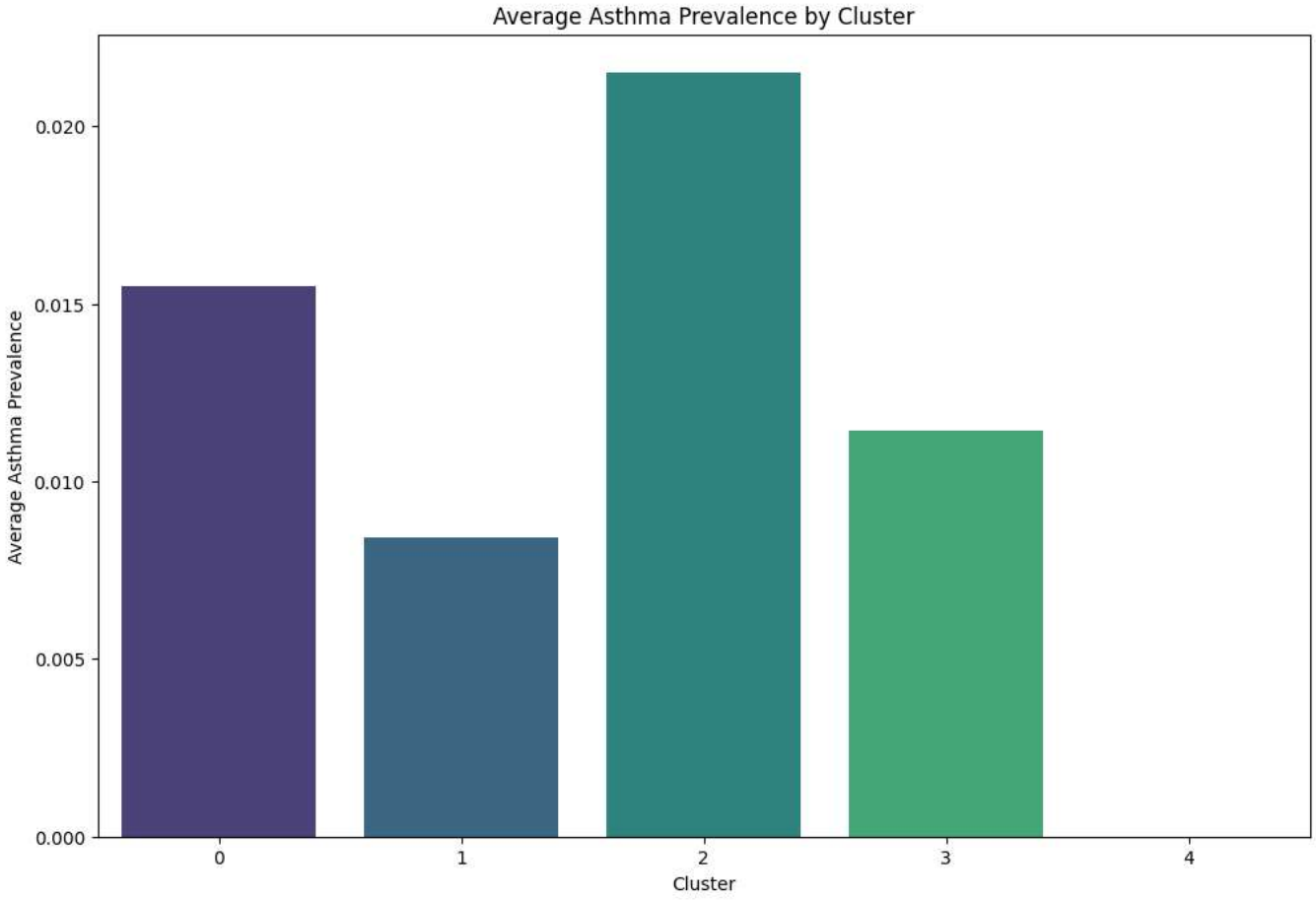
```
Index(['Age', 'Gender', 'Race', 'Blood Group', 'Temperature',
       'Blood Pressure (Systolic)', 'Blood Pressure (Diastolic)', 'Height',
       'Weight', 'Needs Medical Attention', 'Numeric Age Group',
       'Number of Health Concerns', 'Asthma', 'Allergies', ' Depression',
       'Diabetes', ' Sexual health education', ' Heart health monitoring',
       'Chronic diseases management', 'Anxiety', ' Diabetes', 'Obesity',
       'Depression', ' Mental health challenges', ' Growth and development',
       ' Obesity', ' Anxiety', ' Hypertension', ' Allergies', 'Vaccination',
       'Unknown', ' Diet and exercise counseling', 'Acne and skin issues',
       'General Checkup', 'Hypertension', ' Asthma', 'Cluster'],
      dtype='object')
```

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

# Select relevant columns (health concerns and other features)
selected_columns = ['Asthma', 'Allergies', 'Depression', 'Diabetes', 'Hypertension', 'Obesity', 'Anxiety', 'Cluster']

# Create a subset of the DataFrame with selected columns
cluster_data = df_medical_attention[selected_columns]

# Group the data by cluster and calculate the mean of each feature
cluster_means = cluster_data.groupby('Cluster').mean().reset_index()

# Plot the distribution of health concerns and other features across clusters
plt.figure(figsize=(12, 8))
sns.barplot(data=cluster_means, x='Cluster', y='Asthma', palette='viridis')
plt.title('Average Asthma Prevalence by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Average Asthma Prevalence')
plt.show()

# List of health concerns and other features
health_concerns = ['Asthma', 'Allergies', 'Depression', 'Diabetes', 'Hypertension', 'Obesity', 'Anxiety']

# Loop through each health concern and feature to create bar plots
for concern in health_concerns:
    plt.figure(figsize=(12, 8))
    sns.barplot(data=cluster_means, x='Cluster', y=concern, palette='viridis')
    plt.title(f'Average {concern} Prevalence by Cluster')
    plt.xlabel('Cluster')
    plt.ylabel(f'Average {concern} Prevalence')
    plt.show()
```

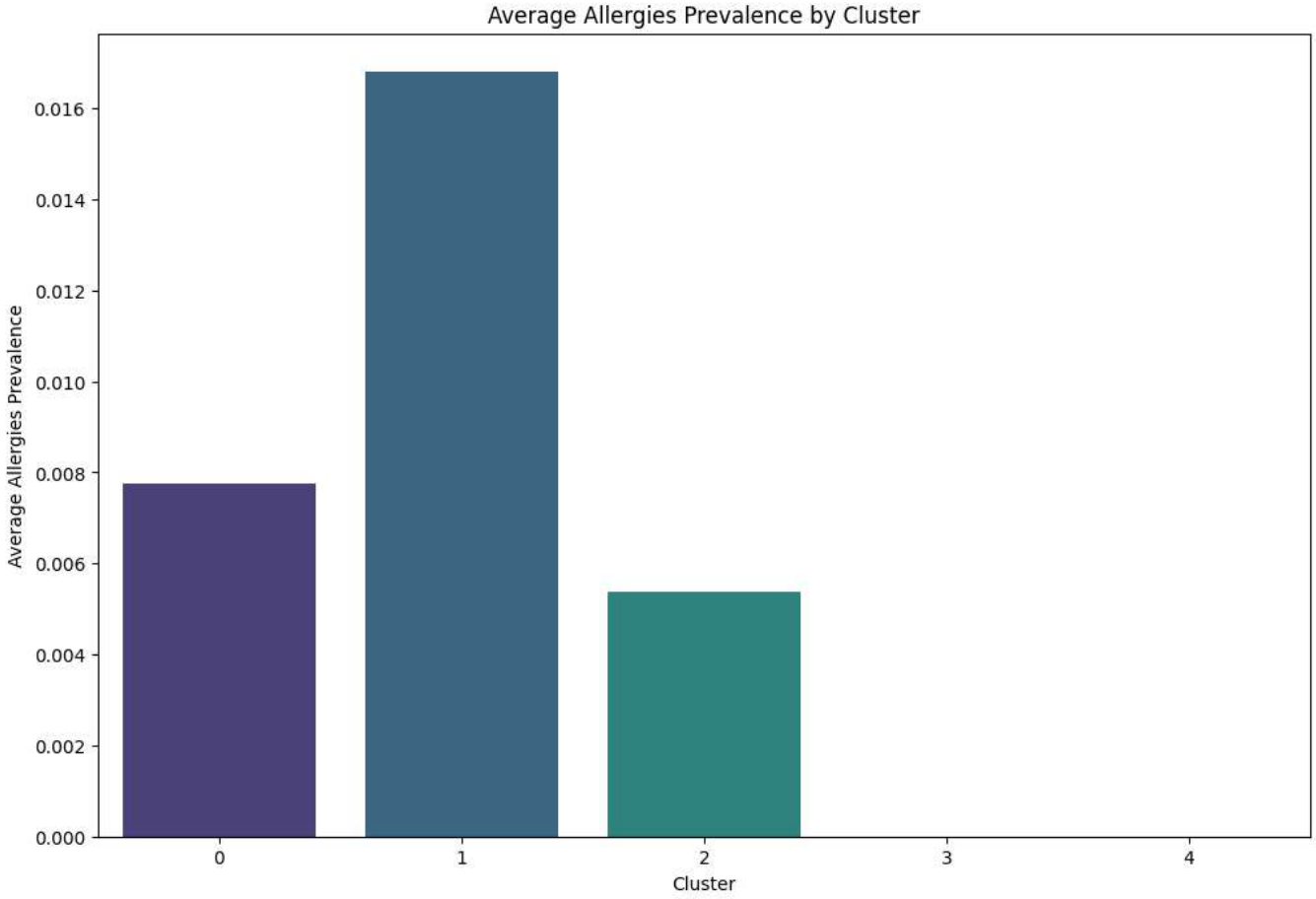
```
<ipython-input-99-2d66d7925f27>:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(data=cluster_means, x='Cluster', y='Asthma', palette='viridis')
```



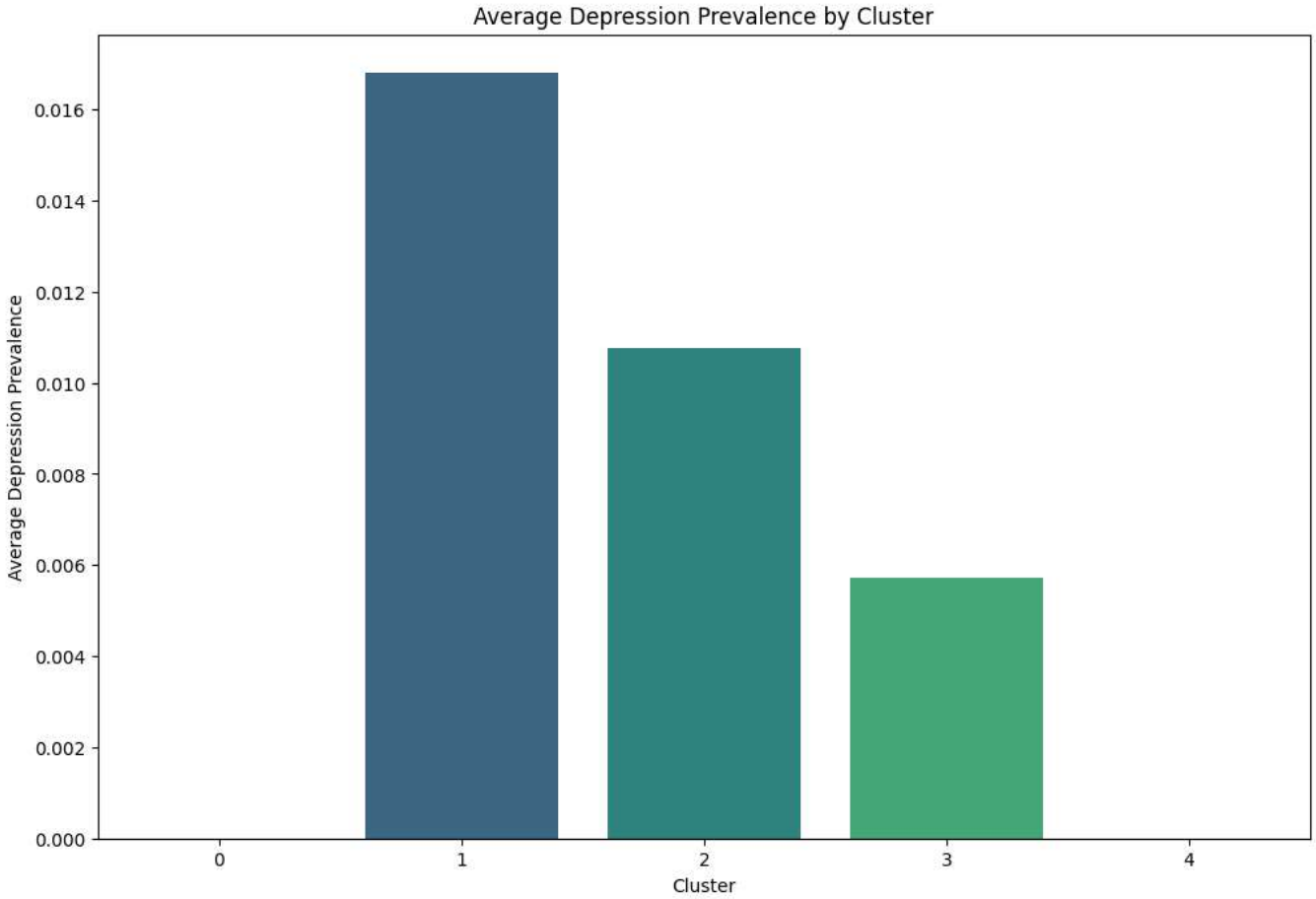

```
<ipython-input-99-2d66d7925f27>:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(data=cluster_means, x='Cluster', y=concern, palette='viridis')
```



```
<ipython-input-99-2d66d7925f27>:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(data=cluster_means, x='Cluster', y=concern, palette='viridis')
```

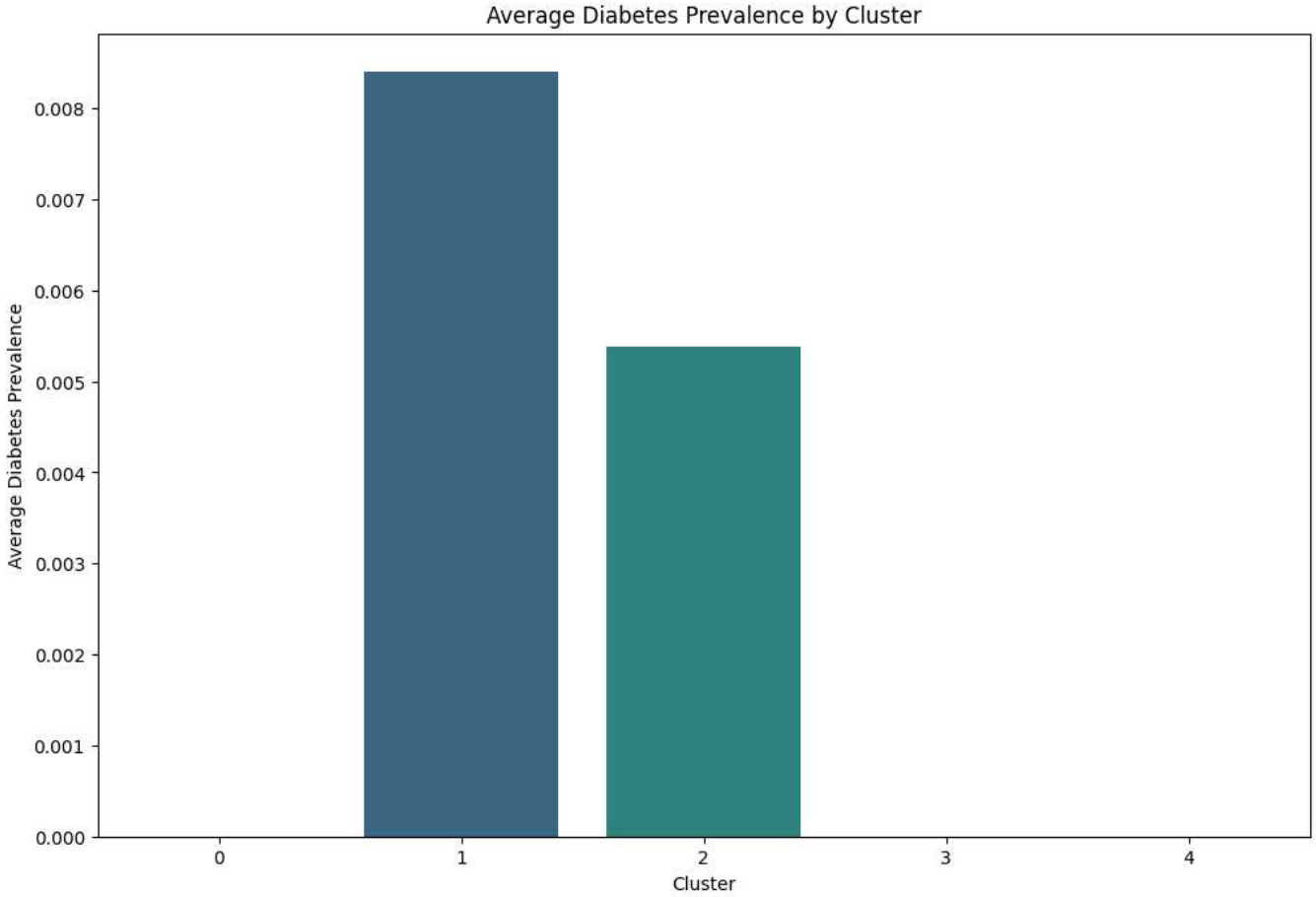



```
<ipython-input-99-2d66d7925f27>:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(data=cluster_means, x='Cluster', y=concern, palette='viridis')
```




```
<ipython-input-99-2d66d7925f27>:27: FutureWarning:
```

```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(data=cluster_means, x='Cluster', y=concern, palette='viridis')
```




```
<ipython-input-99-2d66d7925f27>:27: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `l

  sns.barplot(data=cluster_means, x='Cluster', y=concern, palette='viridis')
```