```
!pip install faker
```

```
Collecting faker
  Downloading Faker-23.2.1-py3-none-any.whl (1.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 10.1 MB/s eta 0:00:00
Requirement already satisfied: python-dateutil>=2.4 in /usr/local/lib/python3.10/dist
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (f
Installing collected packages: faker
Successfully installed faker-23.2.1
```

```
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
    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

    # 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
    if any(condition in health_concerns for condition in ['Hypertension', 'Diabetes', 'As
        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,
        '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/
```

```
/content/drive/MyDrive/Healthcare
```

```
%ls
```

```
HealthcareDataset.ipynb  medical_data_v2.csv  newmedical_data.csv
medical_data.csv         medical_data_v3.csv
```

```
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 dir
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())
```

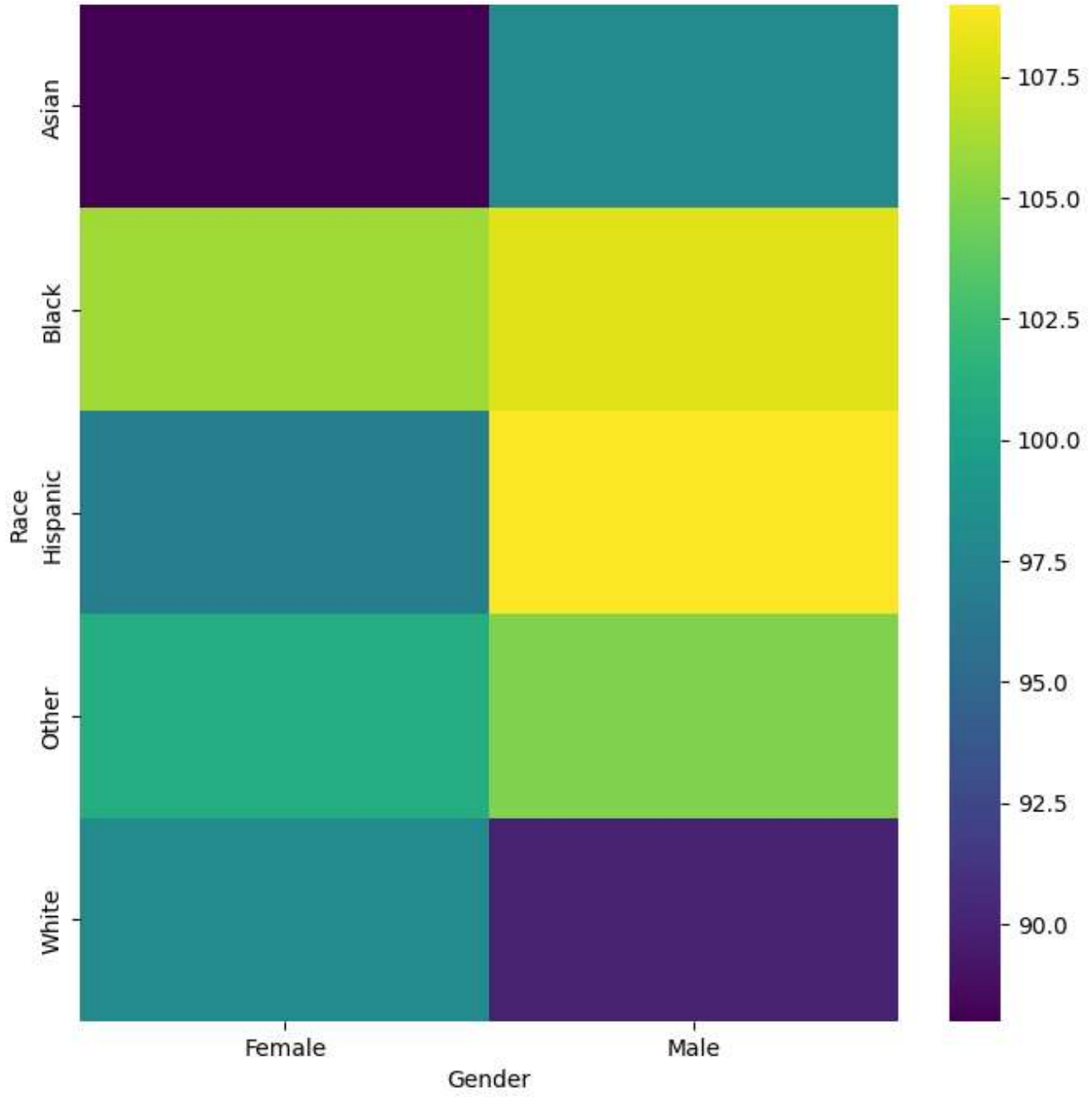
```
First few rows of the dataset:
                 Name   Age Age Group  Gender      Race Blood Group  \
0          Kevin Wong  29.0     Adult  Female     Black         AB+
1    Mr. David Meyers  84.0     Adult    Male     White          A-
2  Robert Sanchez DDS  26.0     Adult    Male  Hispanic   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

Gender vs Race

Show code

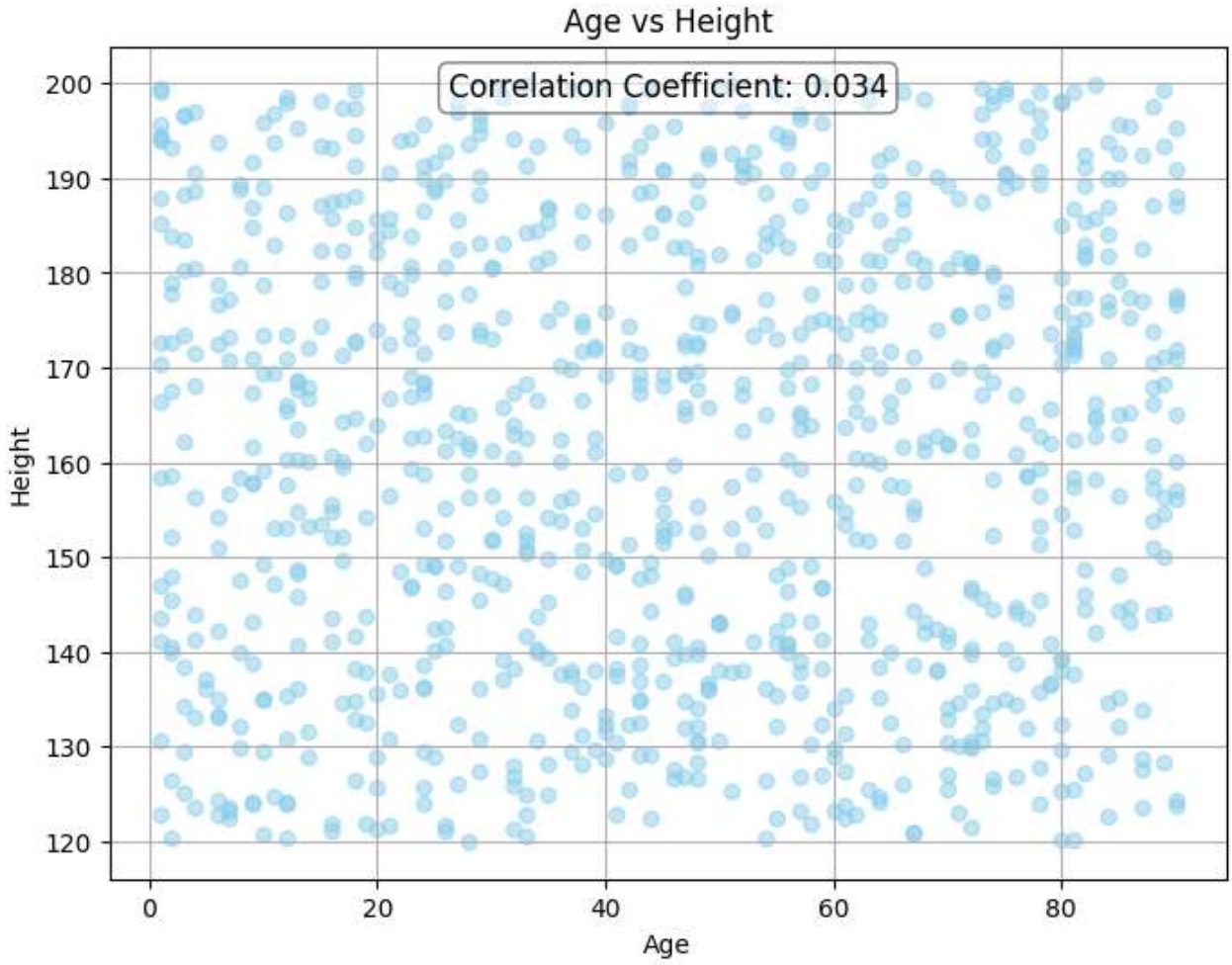


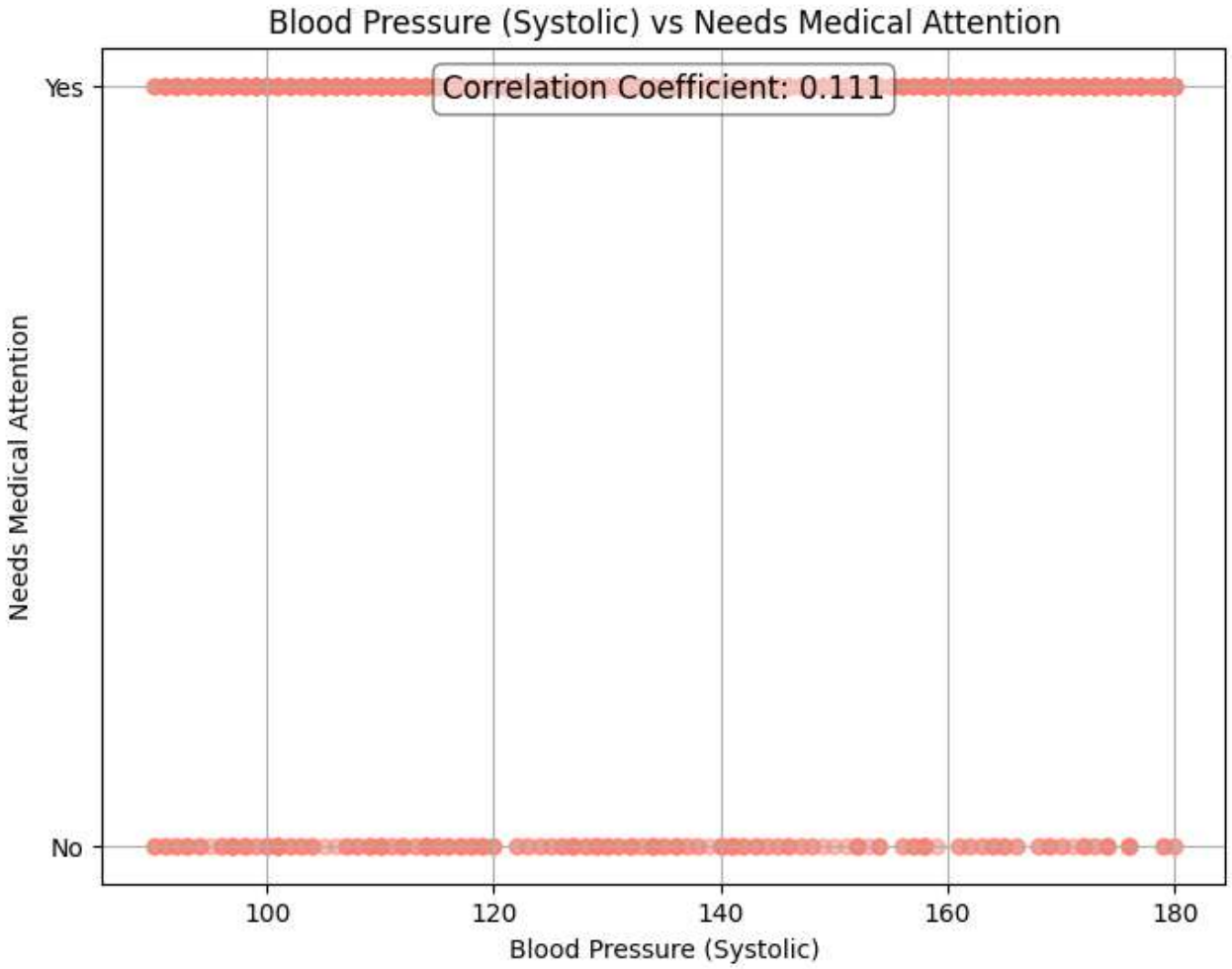

```
# Calculate the correlation matrix
correlation_matrix = df.corr()

# Display the correlation matrix
print("Correlation Matrix:")
print(correlation_matrix)
```

```
Correlation Matrix:
                                Age  Temperature  Blood Pressure (Systolic)  \
Age                        1.000000    -0.043379                  -0.009473
Temperature               -0.043379     1.000000                   0.019296
Blood Pressure (Systolic) -0.009473     0.019296                   1.000000
Blood Pressure (Diastolic)  0.010951    -0.012202                   0.016760
Height                     0.034407     0.029523                   0.042426
Weight                    -0.064587    -0.015252                   0.063763
Needs Medical Attention    0.016633    -0.031977                   0.110760
```

```
                            Blood Pressure (Diastolic)    Height    Weight  \
Age                                           0.010951  0.034407 -0.064587
Temperature                                  -0.012202  0.029523 -0.015252
Blood Pressure (Systolic)                     0.016760  0.042426  0.063763
Blood Pressure (Diastolic)                    1.000000  0.005268  0.005065
Height                                        0.005268  1.000000 -0.001026
Weight                                        0.005065 -0.001026  1.000000
Needs Medical Attention                       0.068096 -0.043843  0.080793

                            Needs Medical Attention
Age                                        0.016633
Temperature                               -0.031977
Blood Pressure (Systolic)                  0.110760
Blood Pressure (Diastolic)                 0.068096
Height                                    -0.043843
Weight                                     0.080793
Needs Medical Attention                    1.000000
<ipython-input-23-22ae92833871>:2: FutureWarning: The default value of numeric_only i
  correlation_matrix = df.corr()
```

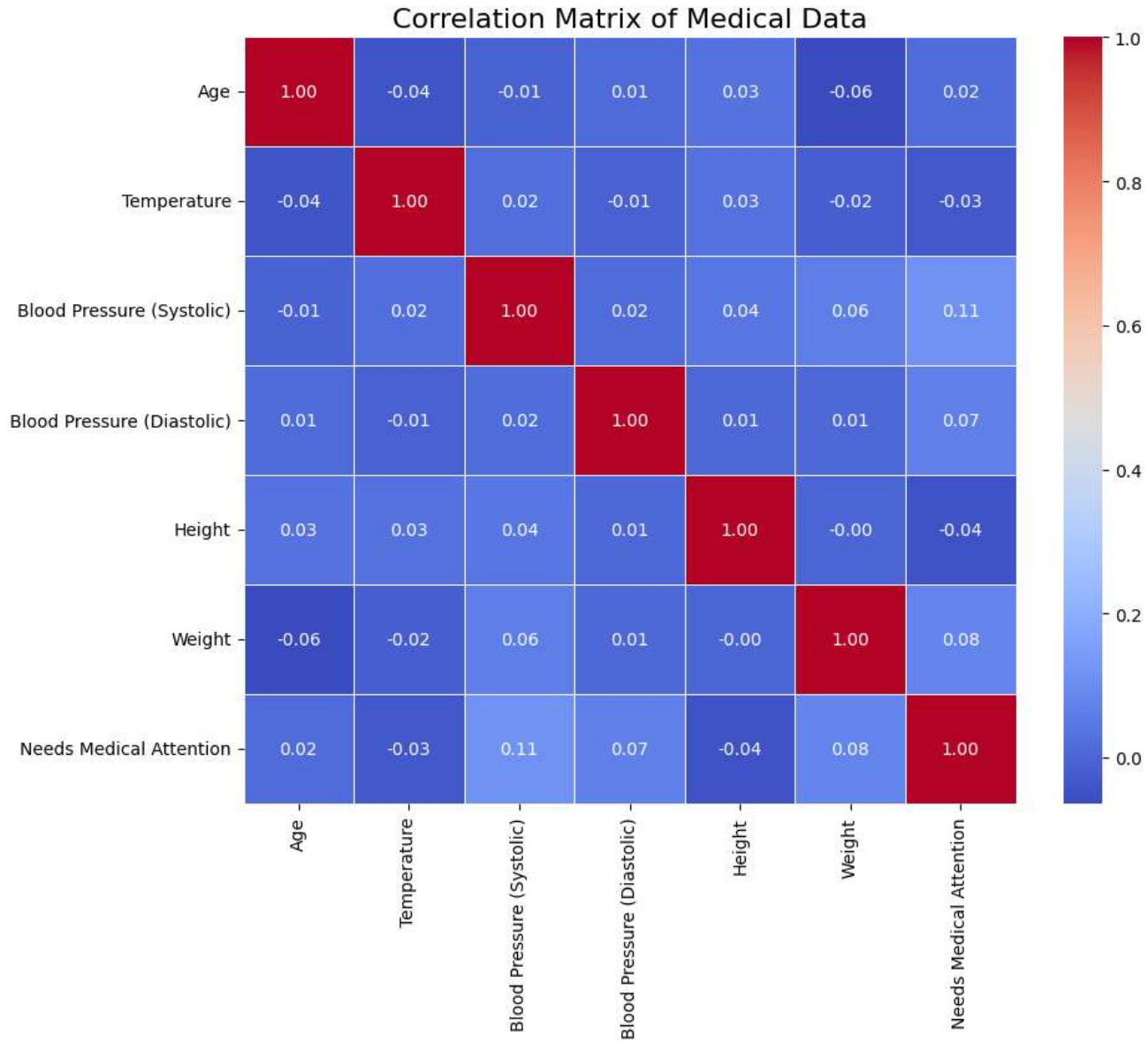
```
import matplotlib.pyplot as plt
import numpy as np

# Scatter plot for Age vs Height
plt.figure(figsize=(8, 6))
plt.scatter(df['Age'], df['Height'], color='skyblue', alpha=0.5)
plt.title('Age vs Height')
plt.xlabel('Age')
plt.ylabel('Height')
plt.grid(True)

# Add correlation coefficient as annotation
corr_coeff = 0.034
plt.annotate(f'Correlation Coefficient: {corr_coeff:.3f}',
             xy=(0.5, 0.95), xycoords='axes fraction',
             fontsize=12, ha='center', va='center',
             bbox=dict(boxstyle='round', facecolor='white', alpha=0.5))

plt.show()

# Scatter plot for Blood Pressure (Systolic) vs Needs Medical Attention
plt.figure(figsize=(8, 6))
plt.scatter(df['Blood Pressure (Systolic)'], df['Needs Medical Attention'], color='salmon
plt.title('Blood Pressure (Systolic) vs Needs Medical Attention')
plt.xlabel('Blood Pressure (Systolic)')
plt.ylabel('Needs Medical Attention')
plt.yticks([0, 1], ['No', 'Yes'])
plt.grid(True)

# Add correlation coefficient as annotation
corr_coeff = 0.111
plt.annotate(f'Correlation Coefficient: {corr_coeff:.3f}',
             xy=(0.5, 0.95), xycoords='axes fraction',
             fontsize=12, ha='center', va='center',
             bbox=dict(boxstyle='round', facecolor='white', alpha=0.5))

plt.show()
```


Age vs Height
Correlation Coefficient: 0.034
Height
Age
200
190
180
170
160
150
140
130
120
0
20
40
60
80


Blood Pressure (Systolic) vs Needs Medical Attention
Correlation Coefficient: 0.111
Needs Medical Attention
Blood Pressure (Systolic)
Yes
No
100
120
140
160
180

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

# Calculate the correlation matrix
correlation_matrix = df.corr()

# Set up the matplotlib figure
plt.figure(figsize=(10, 8))

# Plot the heatmap
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

# Add title and rotate y-axis labels
plt.title('Correlation Matrix of Medical Data', fontsize=16)
plt.yticks(rotation=0)
plt.xticks(rotation=90)

# Display the heatmap
plt.show()
```

```
<ipython-input-24-8b374aa8af03>:5: FutureWarning: The default value of numeric_only i
  correlation_matrix = df.corr()
```




```
# Assuming df is DataFrame
column_datatypes = df.dtypes
print(column_datatypes)
```

```
Name                           object
Age                           float64
Age Group                      object
Gender                         object
Race                           object
Blood Group                    object
Temperature                   float64
Blood Pressure (Systolic)       int64
Blood Pressure (Diastolic)      int64
Height                        float64
Weight                        float64
Health Concerns                object
Needs Medical Attention         int64
dtype: object
```

```
quantitative_continuous = []  # List to store names of columns with continuous quantitati
quantitative_discrete = []    # List to store names of columns with discrete quantitative
qualitative_nominal = []      # List to store names of columns with nominal qualitative d
qualitative_ordinal = []      # List to store names of columns with ordinal qualitative d

for column in df.columns:
    if df[column].dtype in ['int64', 'float64']:  # Check if the data type is numerical
        if df[column].nunique() > 10:  # Arbitrary threshold to differentiate between con
            quantitative_continuous.append(column)
        else:
            quantitative_discrete.append(column)
    else:
        if df[column].nunique() > 10:  # Arbitrary threshold to differentiate between nom
            qualitative_nominal.append(column)
        else:
            qualitative_ordinal.append(column)

print("Quantitative Continuous:", quantitative_continuous)
print("Quantitative Discrete:", quantitative_discrete)
print("Qualitative Nominal:", qualitative_nominal)
print("Qualitative Ordinal:", qualitative_ordinal)
```

```
Quantitative Continuous: ['Age', 'Temperature', 'Blood Pressure (Systolic)', 'Blood P
Quantitative Discrete: ['Needs Medical Attention']
Qualitative Nominal: ['Name', 'Health Concerns']
Qualitative Ordinal: ['Age Group', 'Gender', 'Race', 'Blood Group']
```

```
import pandas as pd

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

breakdown_df = pd.DataFrame(breakdown_data)

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

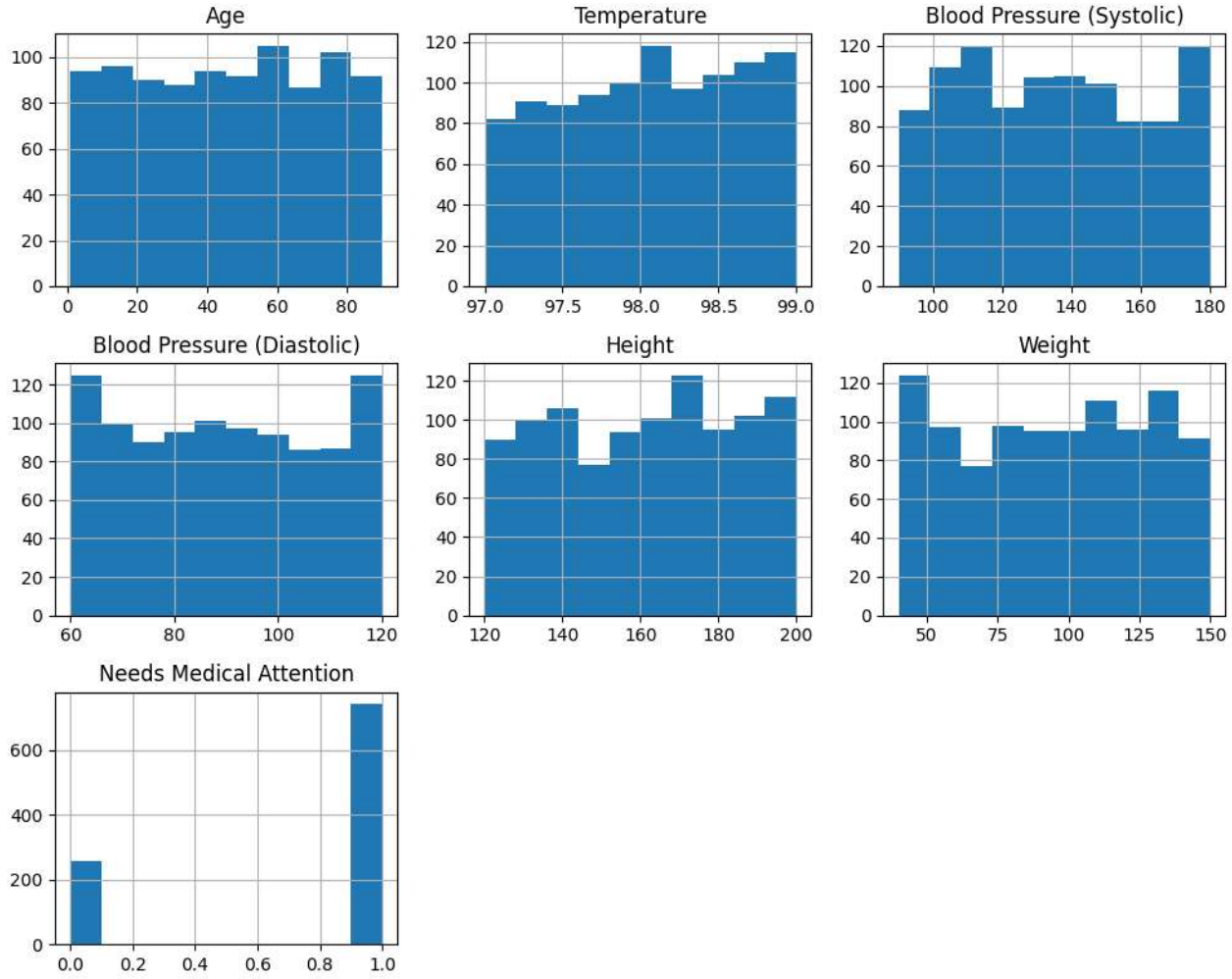
```
Breakdown of Dataset:
                        Column               Data Type
0                         Name                 Nominal
1                          Age  Quantitative, Discrete
2                    Age Group                 Nominal
3                       Gender                 Nominal
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
```

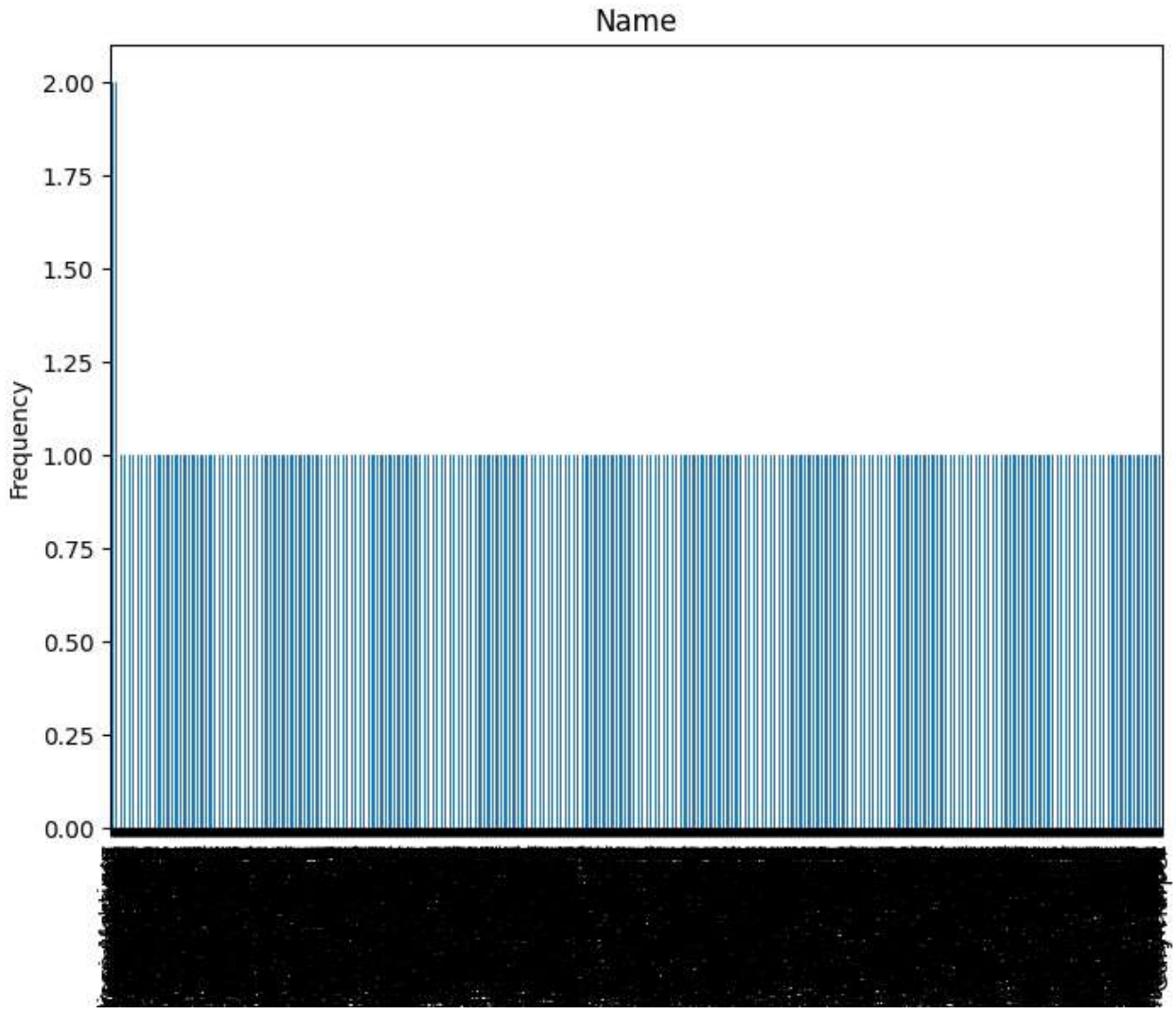
```
print (df)
```

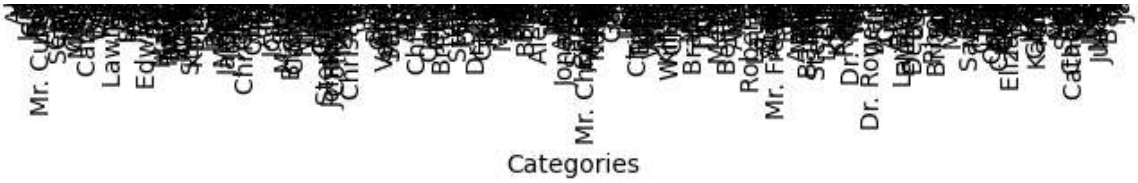
```
                    Name   Age Age Group  Gender      Race Blood Group  \
0            Kevin Wong  29.0     Adult  Female     Black         AB+
1      Mr. David Meyers  84.0     Adult    Male     White          A-
2    Robert Sanchez DDS  26.0     Adult    Male  Hispanic          A+
3           Brent 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
```

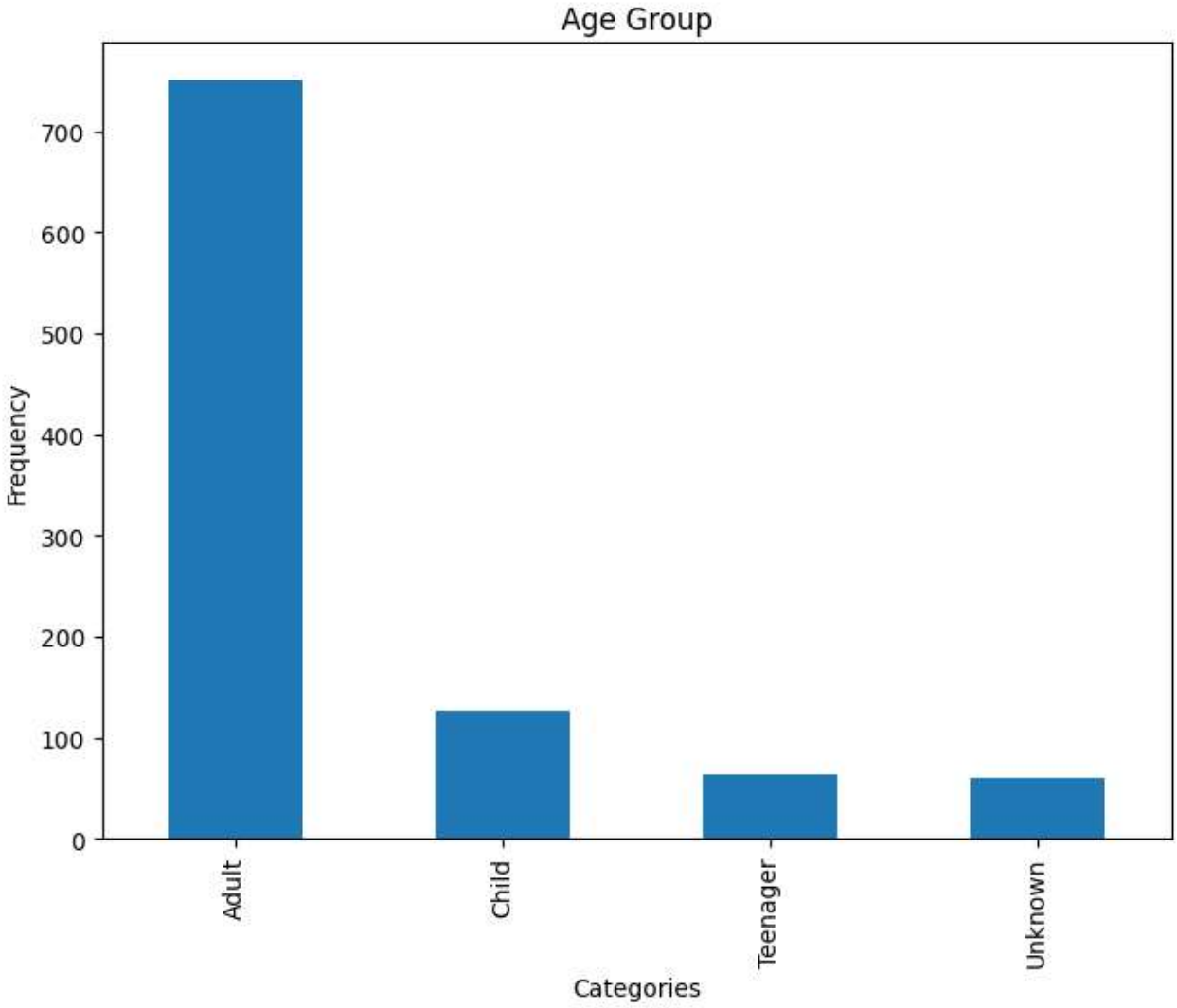
```
998         98.3                          93                          84
999         98.0                         128                          81

     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]
```

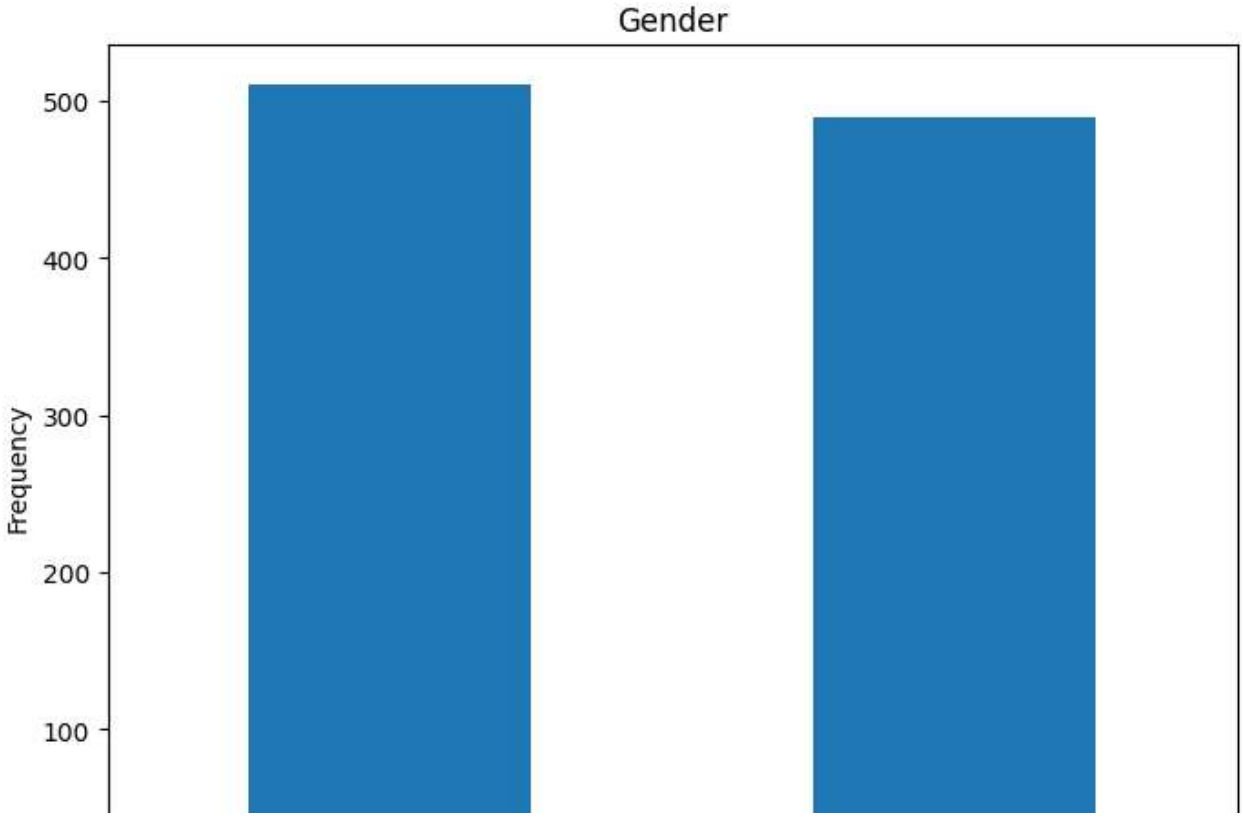
```
import matplotlib.pyplot as plt
import seaborn as sns

# 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

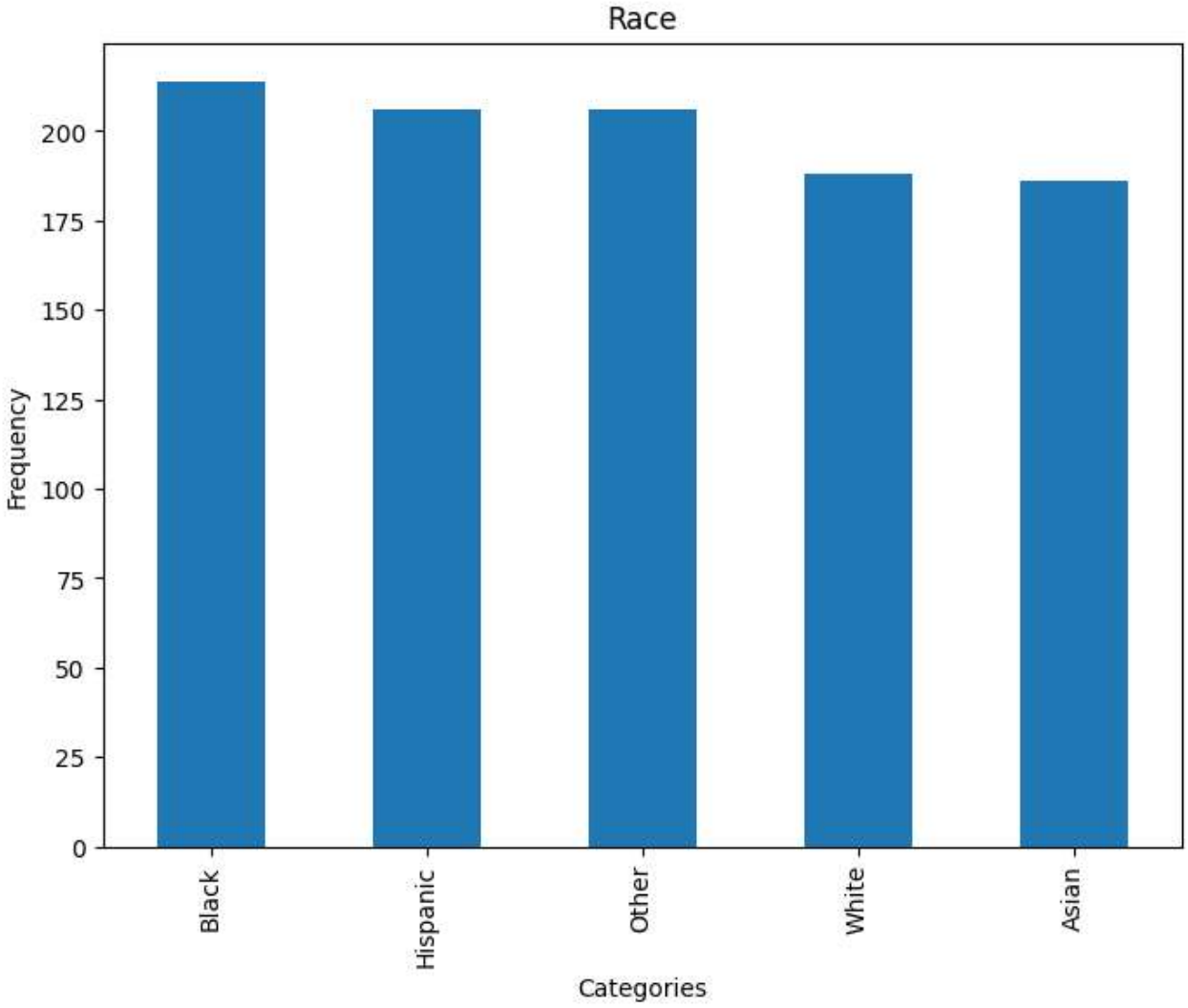

Name
Frequency

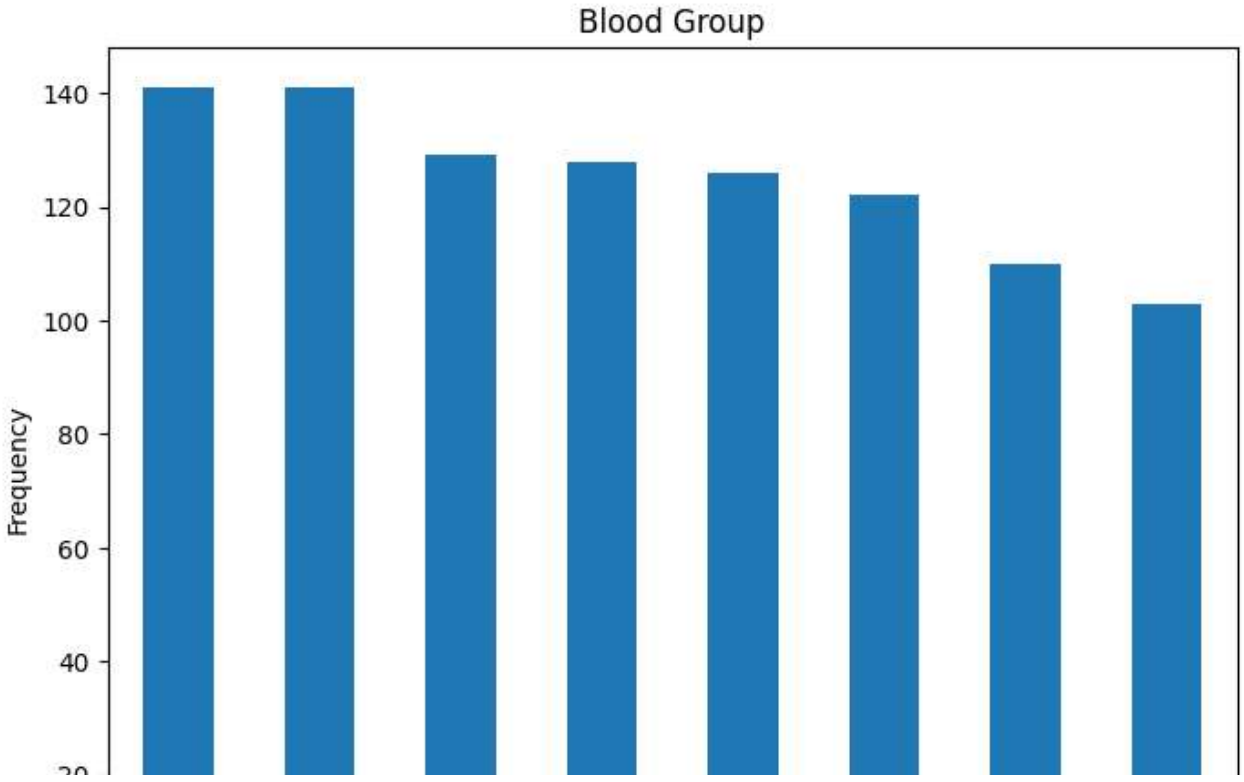

Categories

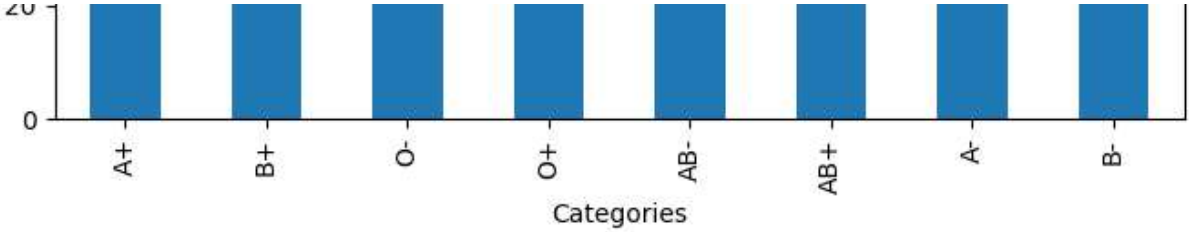

Age Group
Frequency
0
100
200
300
400
500
600
700
Adult
Child
Teenager
Unknown
Categories

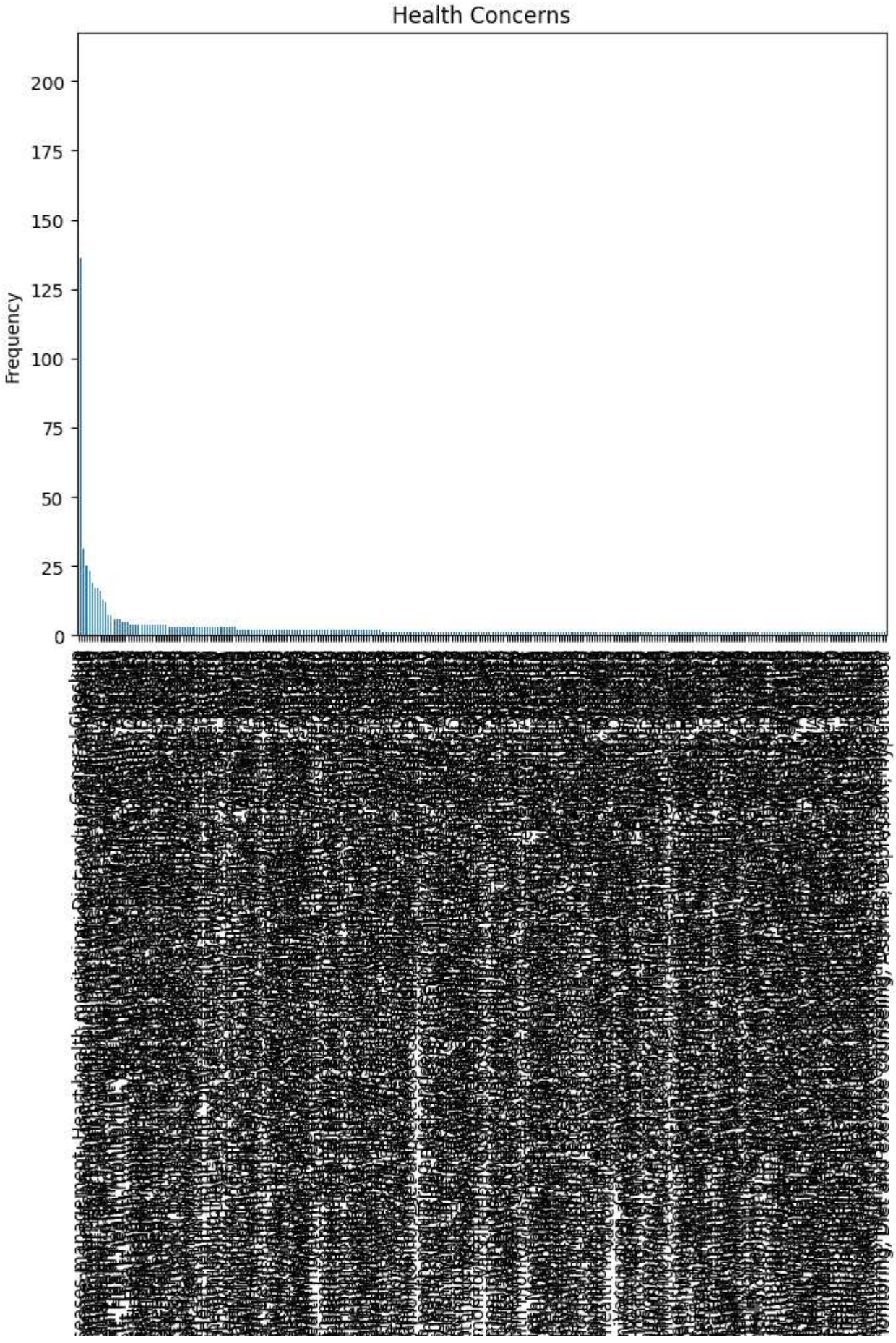

Gender
Frequency
100
200
300
400
500

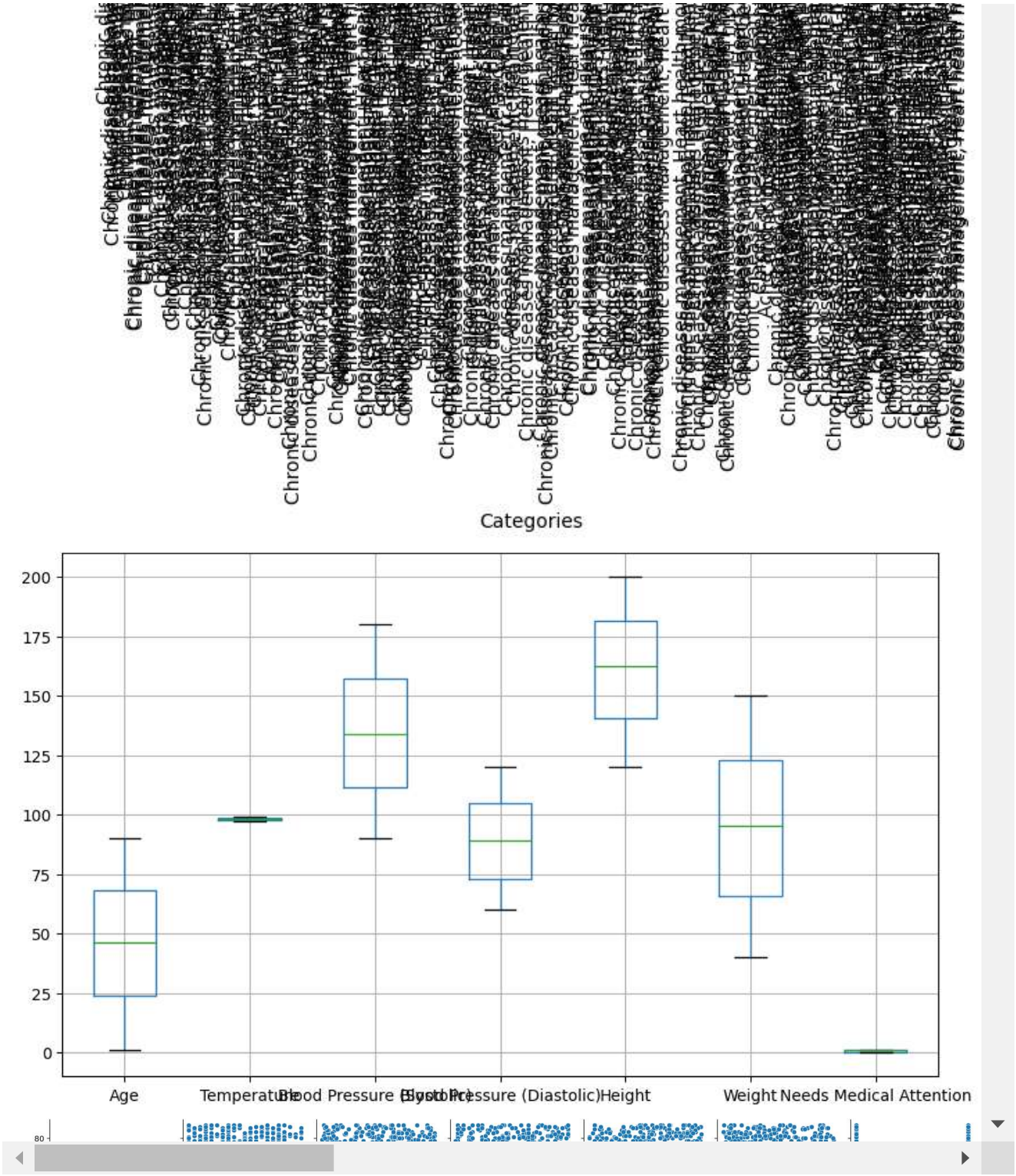

0
Male
Female
Categories


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

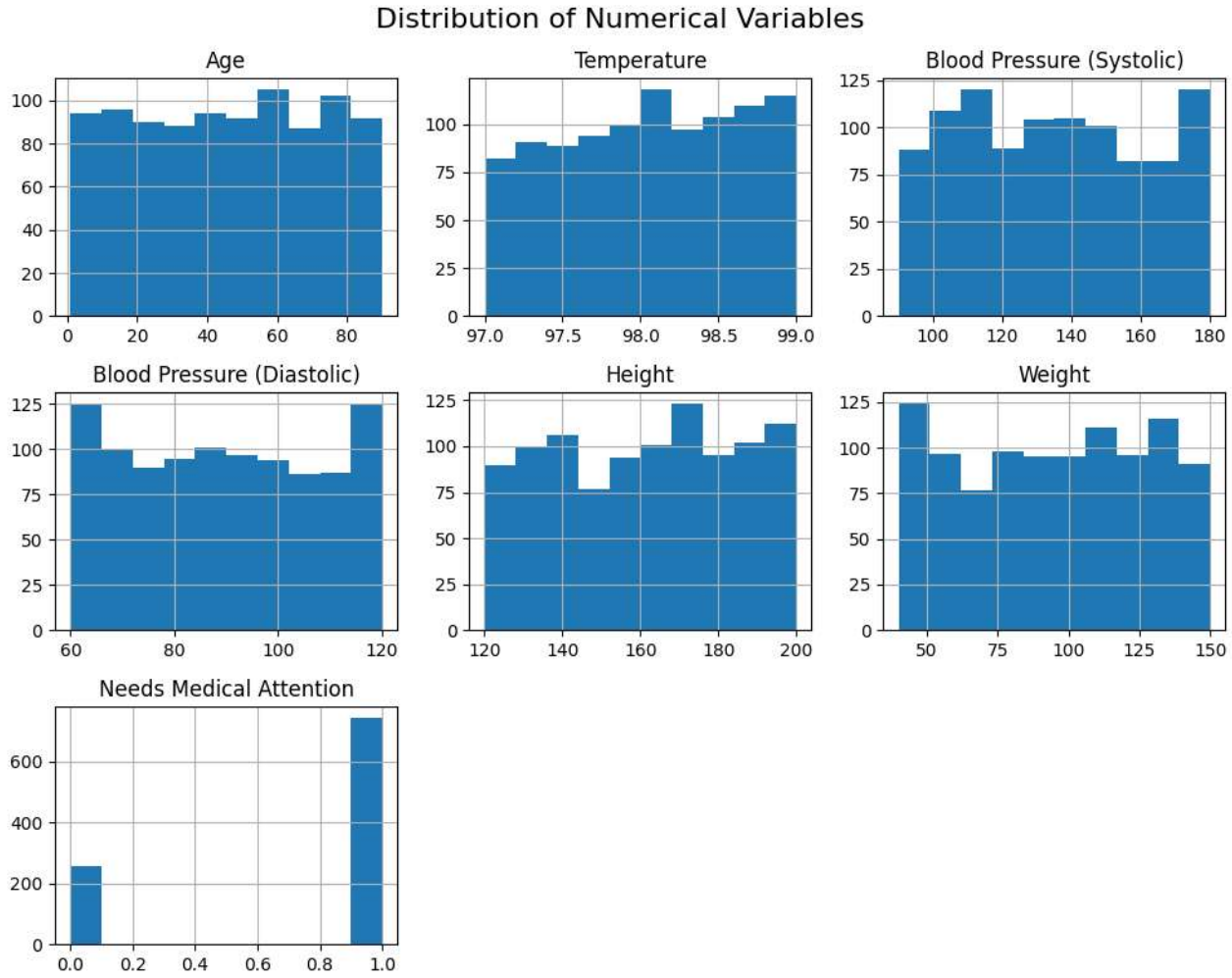

Blood Group
Frequency
140
120
100
80
60
40

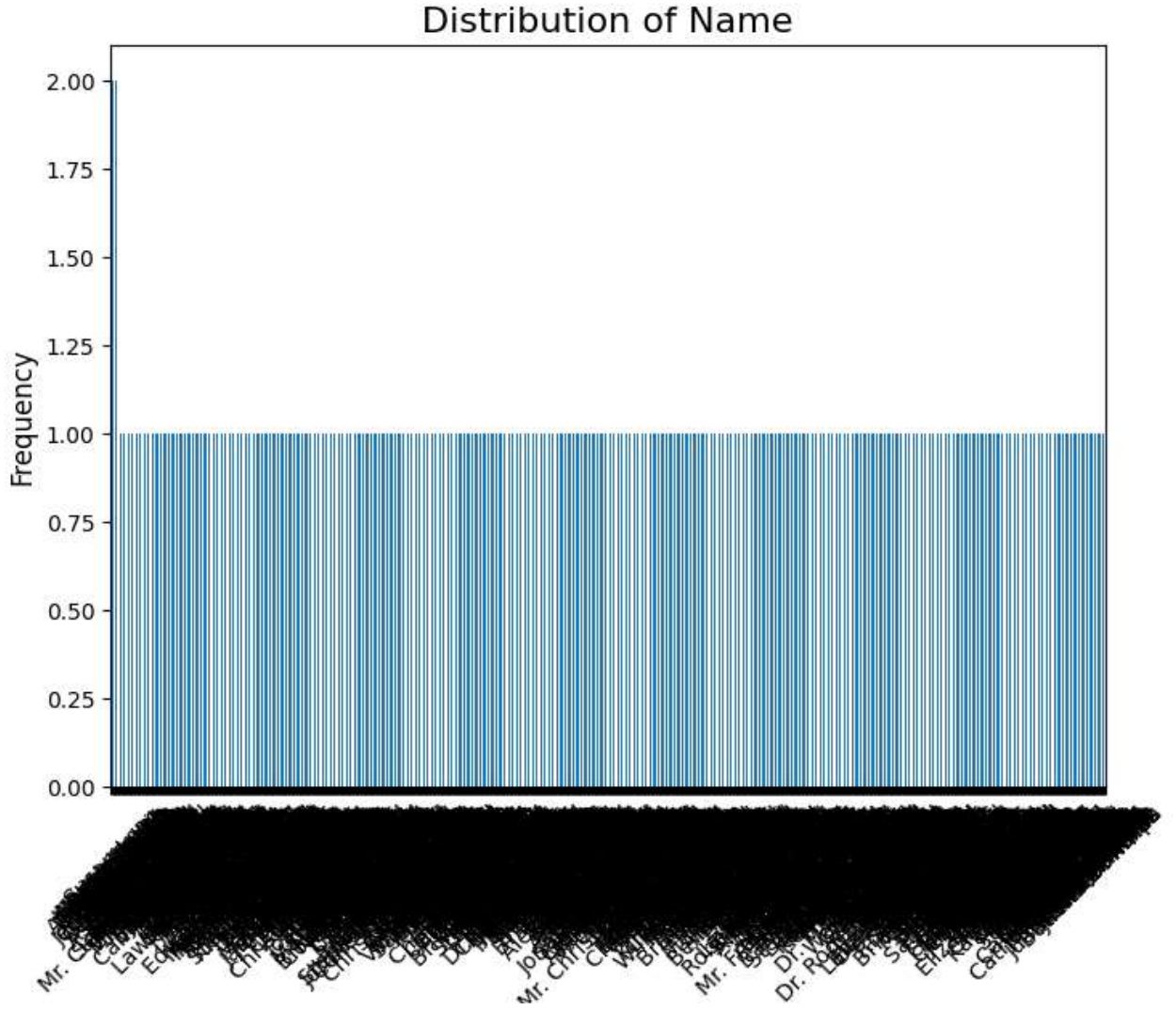

20
0
A+
B+
O-
O+
AB-
AB+
A-
B-
Categories

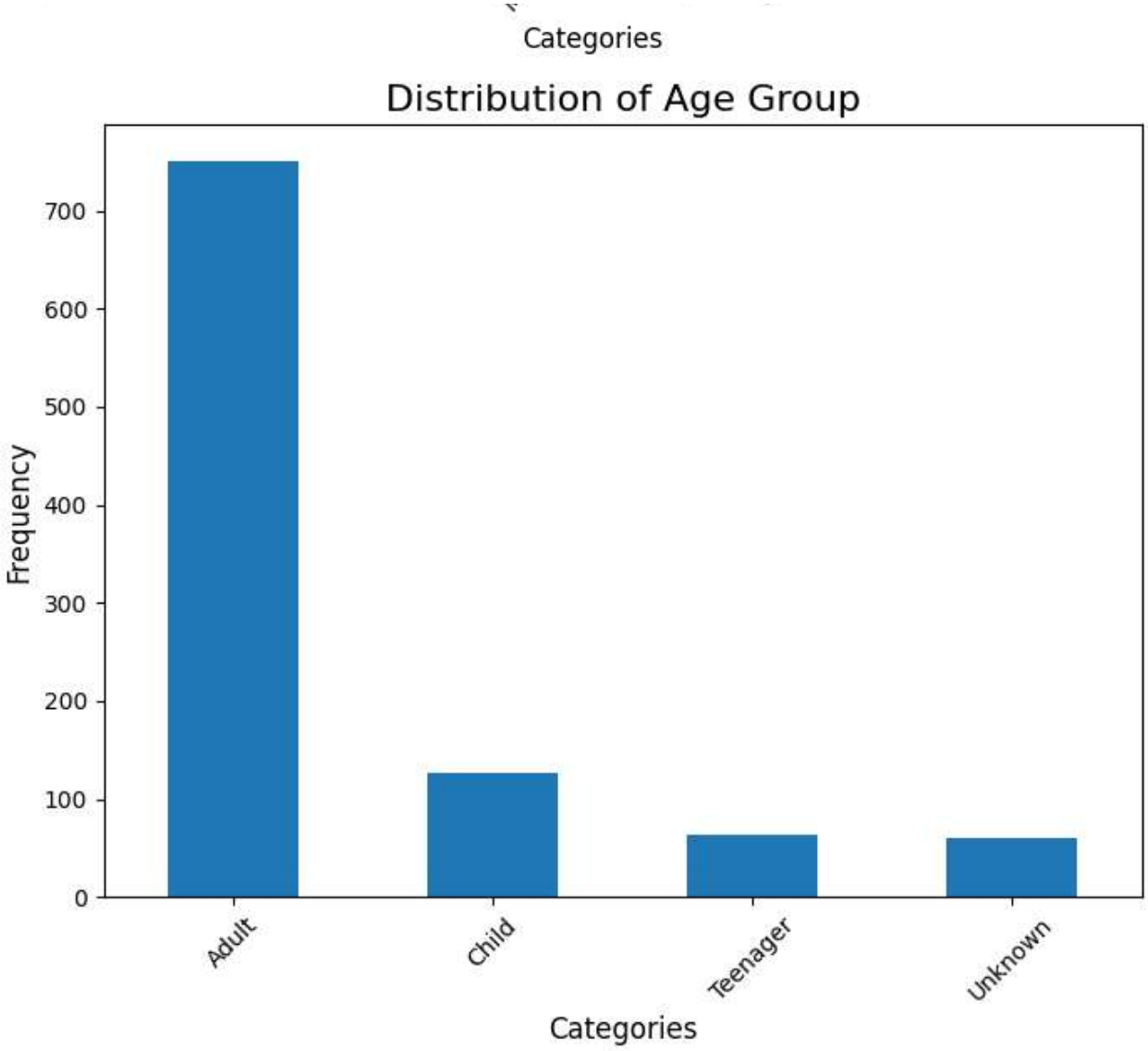

Health Concerns
Frequency
200
175
150
125
100
75
50
25
0

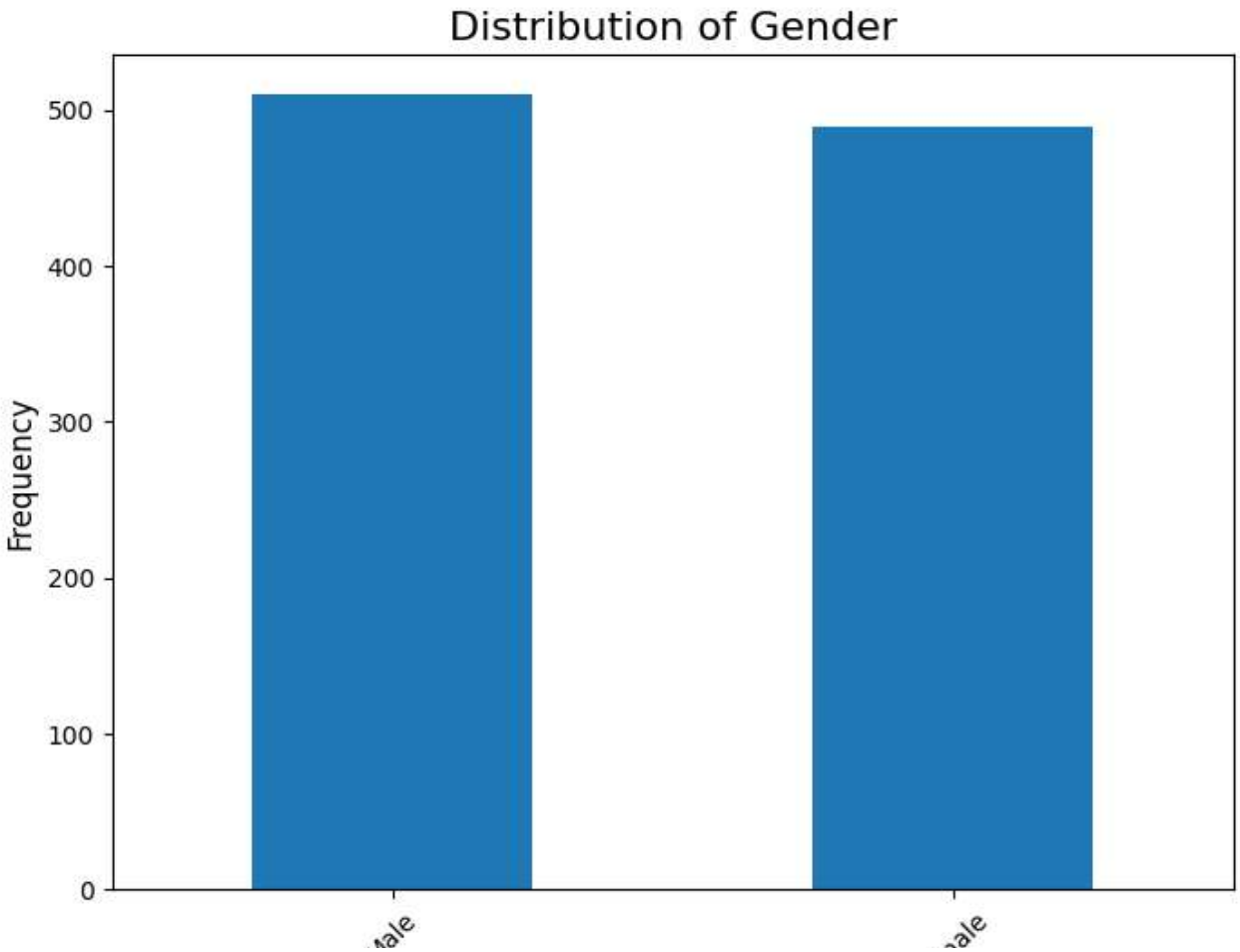

Categories
200
175
150
125
100
75
50
25
0
Age
Temperature
Blood Pressure (Systolic)
Blood Pressure (Diastolic)
Height
Weight
Needs Medical Attention

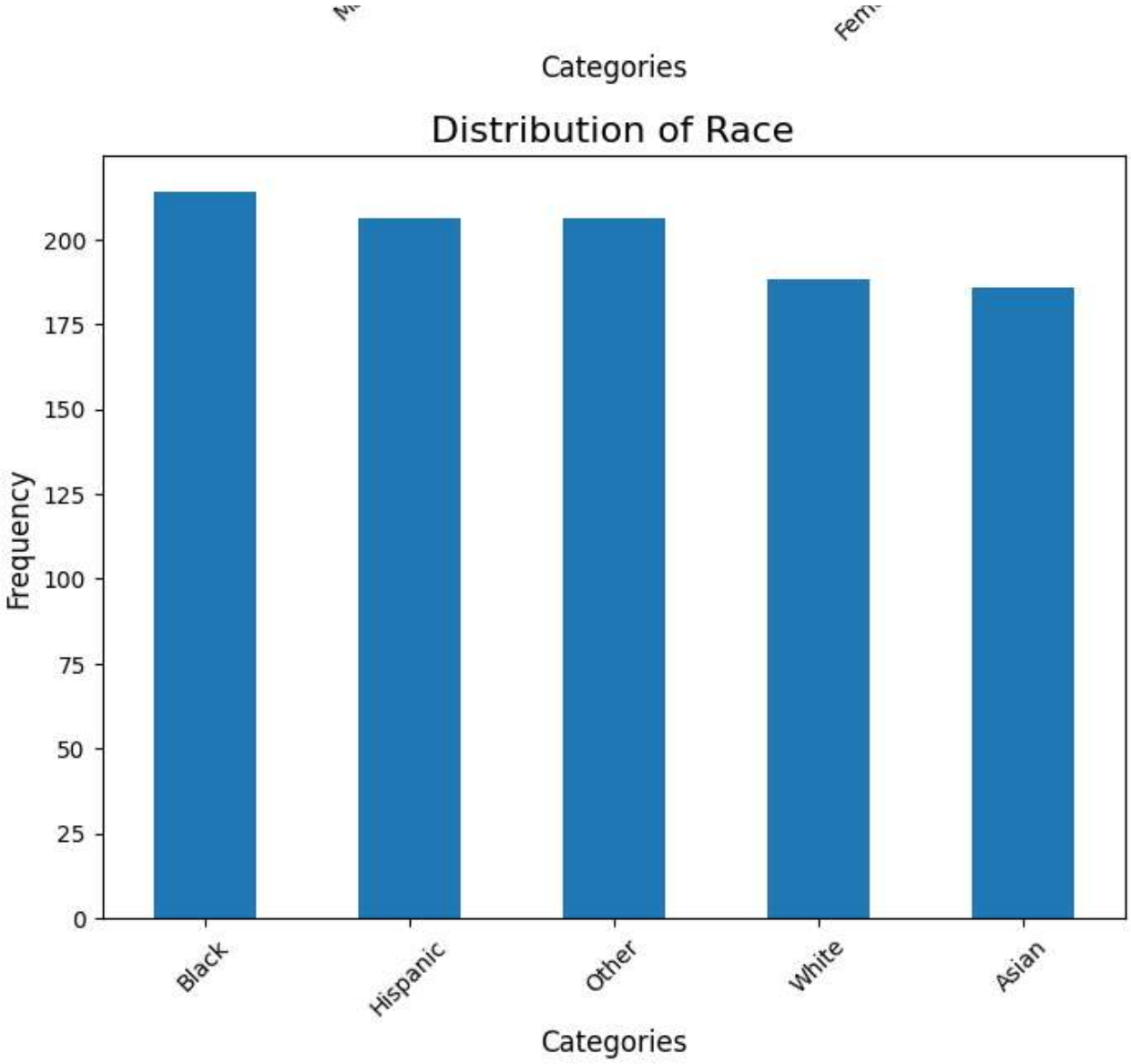
```
import matplotlib.pyplot as plt
import seaborn as sns

# Histograms for numerical variables
df.hist(figsize=(10, 8))
plt.suptitle('Distribution of Numerical Variables', fontsize=16)
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(f'Distribution of {col}', fontsize=16)
    plt.xlabel('Categories', fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.xticks(rotation=45)
    plt.show()

# Box plots for numerical variables
df.boxplot(figsize=(10, 6))
plt.title('Box Plot of Numerical Variables', fontsize=16)
plt.xlabel('Variables', fontsize=12)
plt.ylabel('Values', fontsize=12)
plt.xticks(rotation=45)
plt.show()

# Scatter plots for pairwise relationships
sns.pairplot(df)
plt.suptitle('Pairwise Relationships between Numerical Variables', fontsize=16)
plt.show()
```

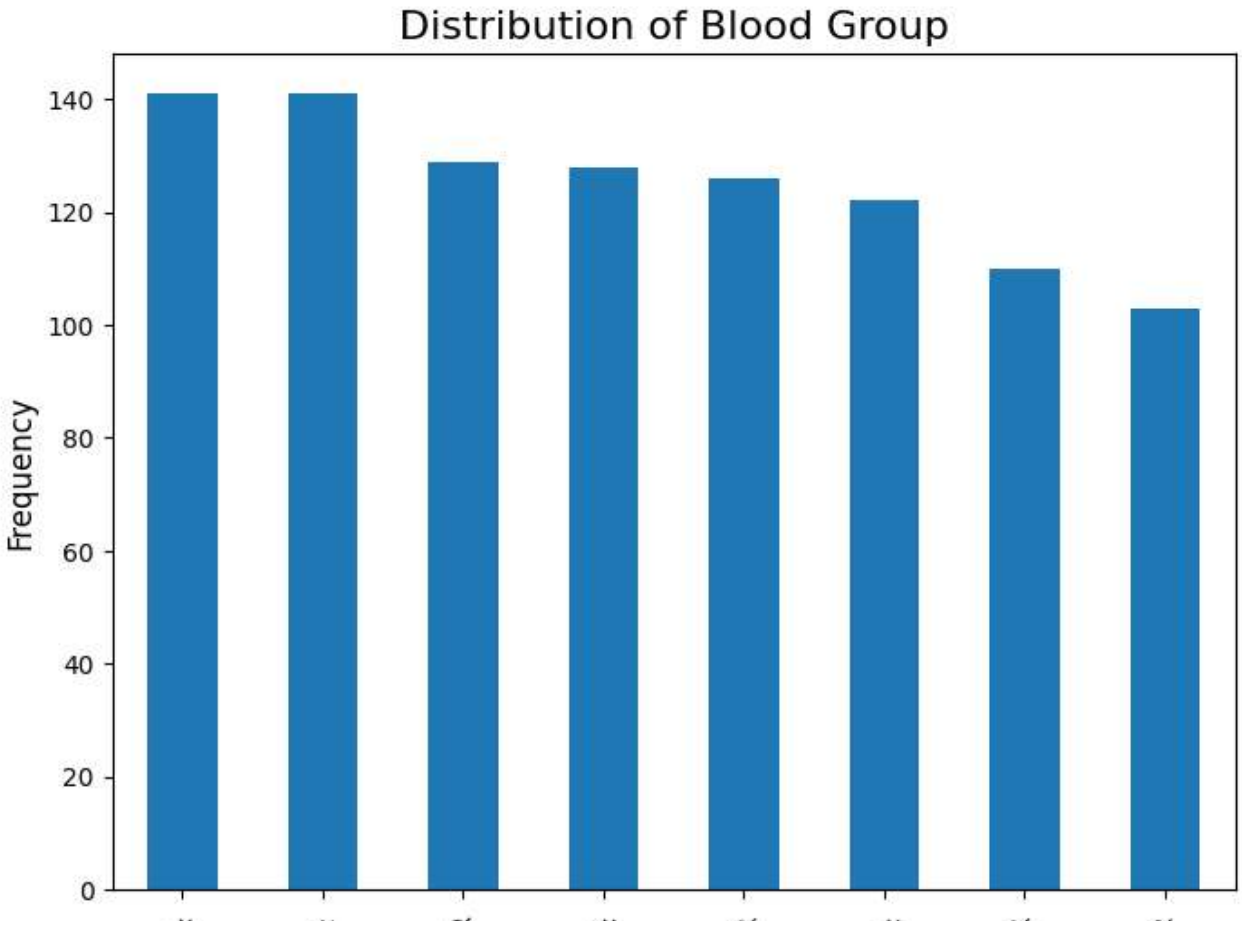

Distribution of Numerical Variables
Age
Temperature
Blood Pressure (Systolic)
Blood Pressure (Diastolic)
Height
Weight
Needs Medical Attention

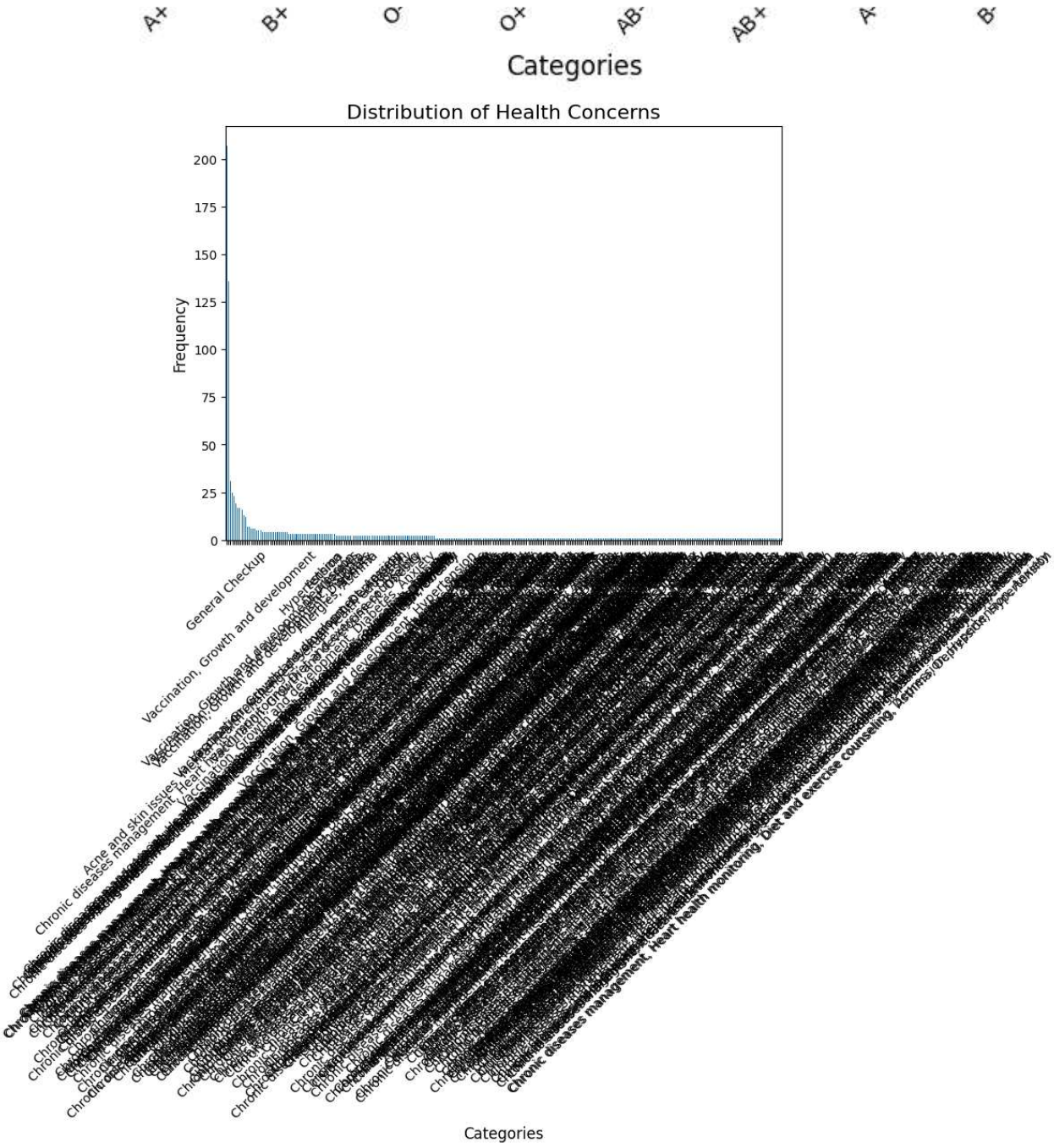

Distribution of Name
Frequency

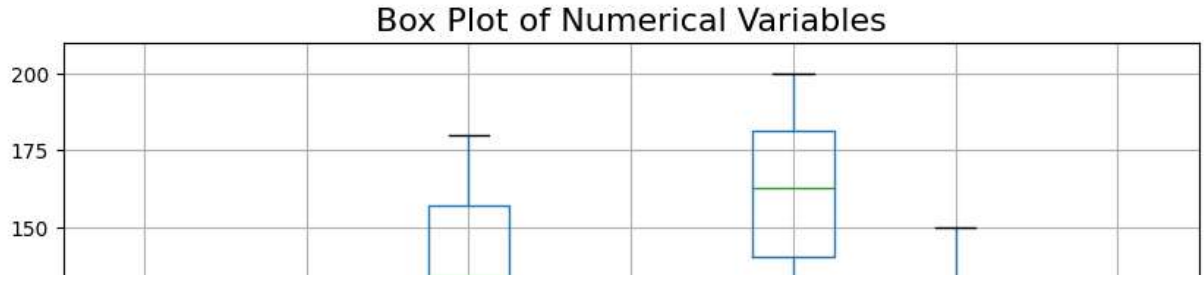

Categories
Distribution of Age Group
Frequency
0
100
200
300
400
500
600
700
Adult
Child
Teenager
Unknown
Categories


Distribution of Gender
Frequency
0
100
200
300
400
500


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


Distribution of Blood Group
Frequency
0
20
40
60
80
100
120
140

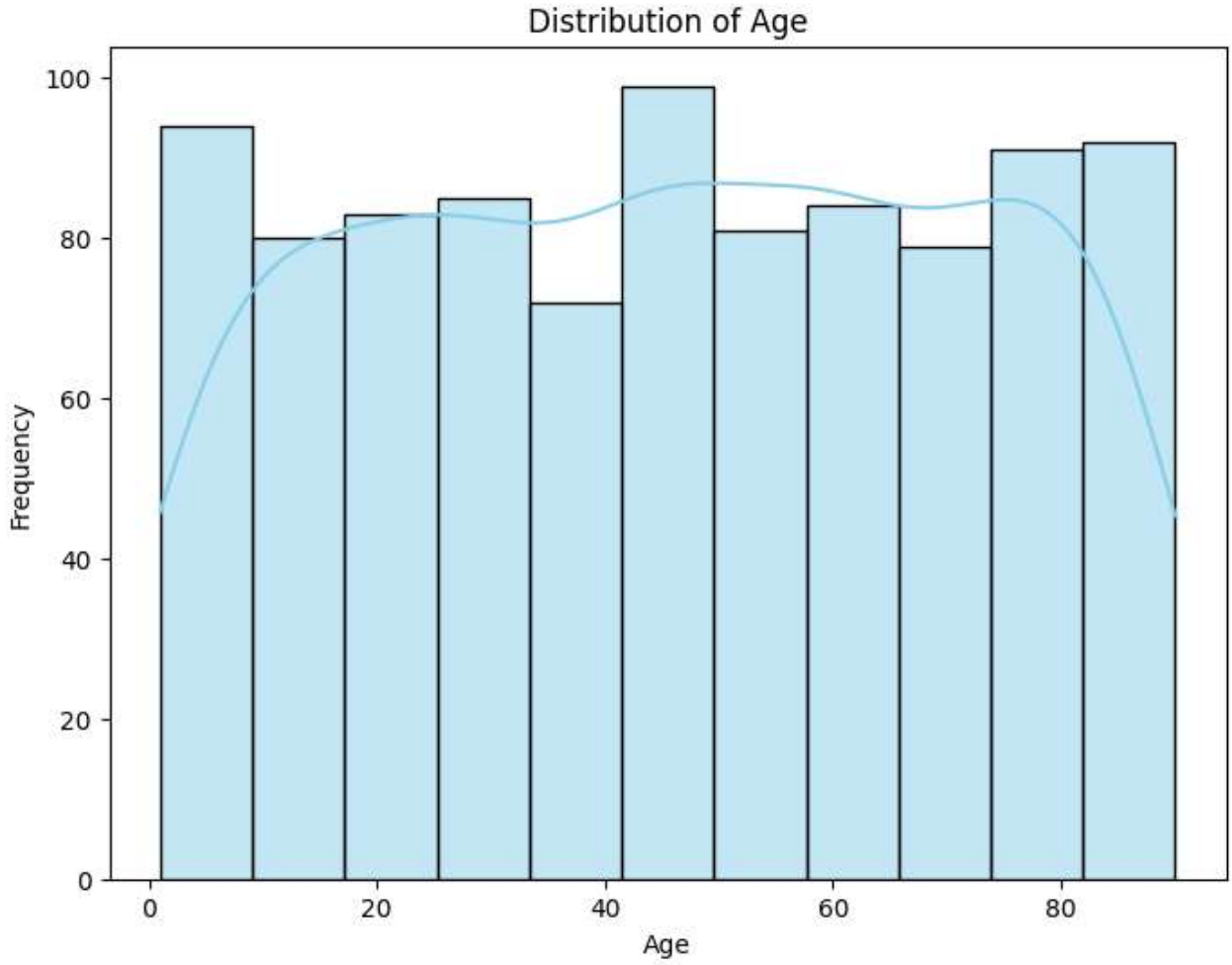

A+
B+
O-
O+
AB-
AB+
A-
B-
Categories
Distribution of Health Concerns
Frequency
200
175
150
125
100
75
50
25
0
General Checkup
Vaccination, Growth and development
Acne and skin issues
Chronic diseases management, Heart health monitoring, Diet and exercise counseling, Distress/Depression/Hypertension
Categories

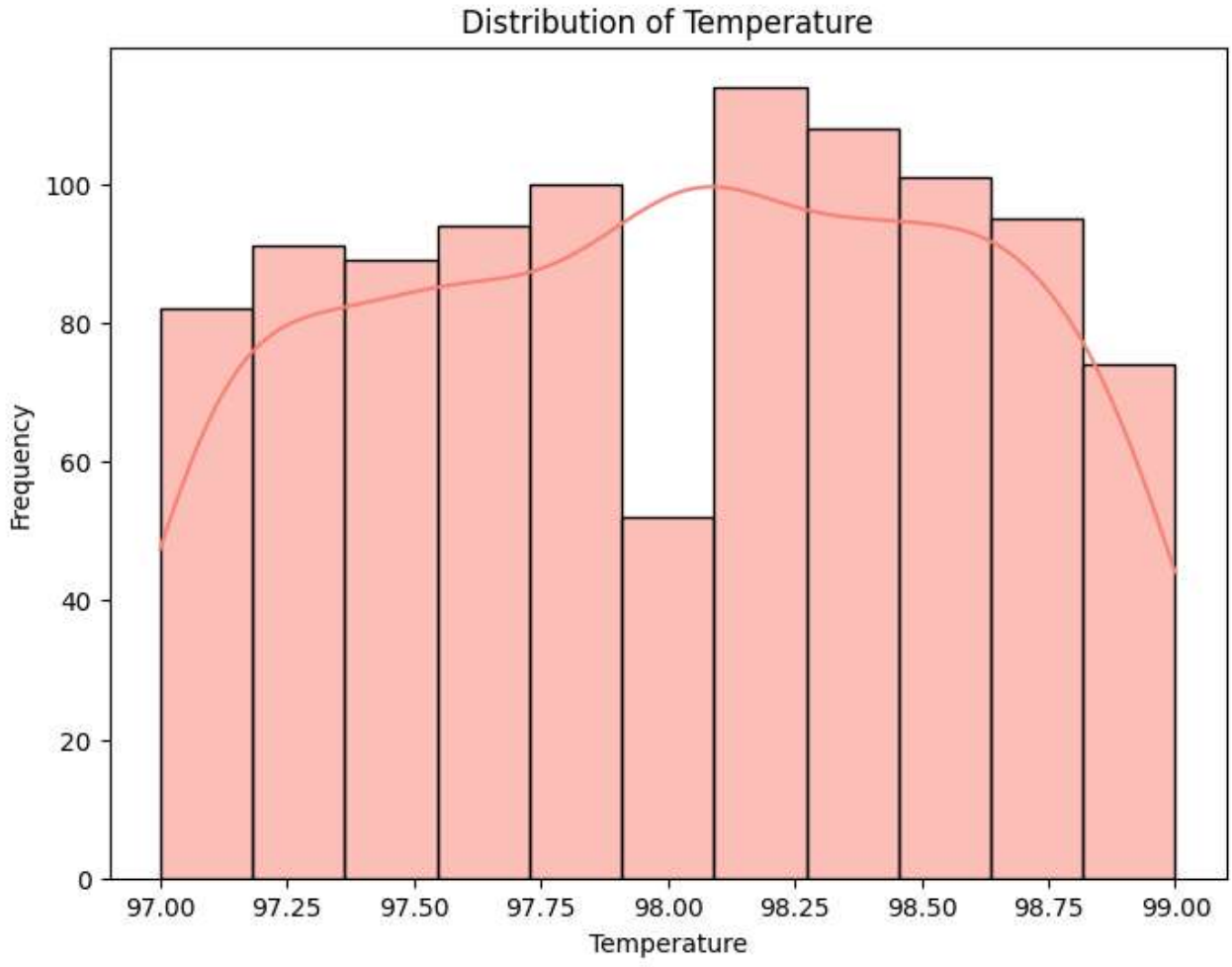

Box Plot of Numerical Variables
200
175
150

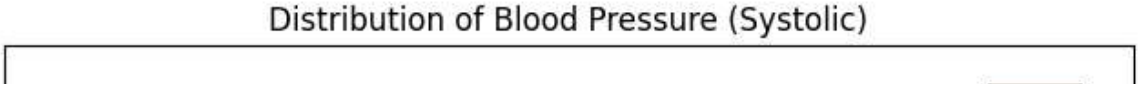
```
import pandas as pd

# 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)
```

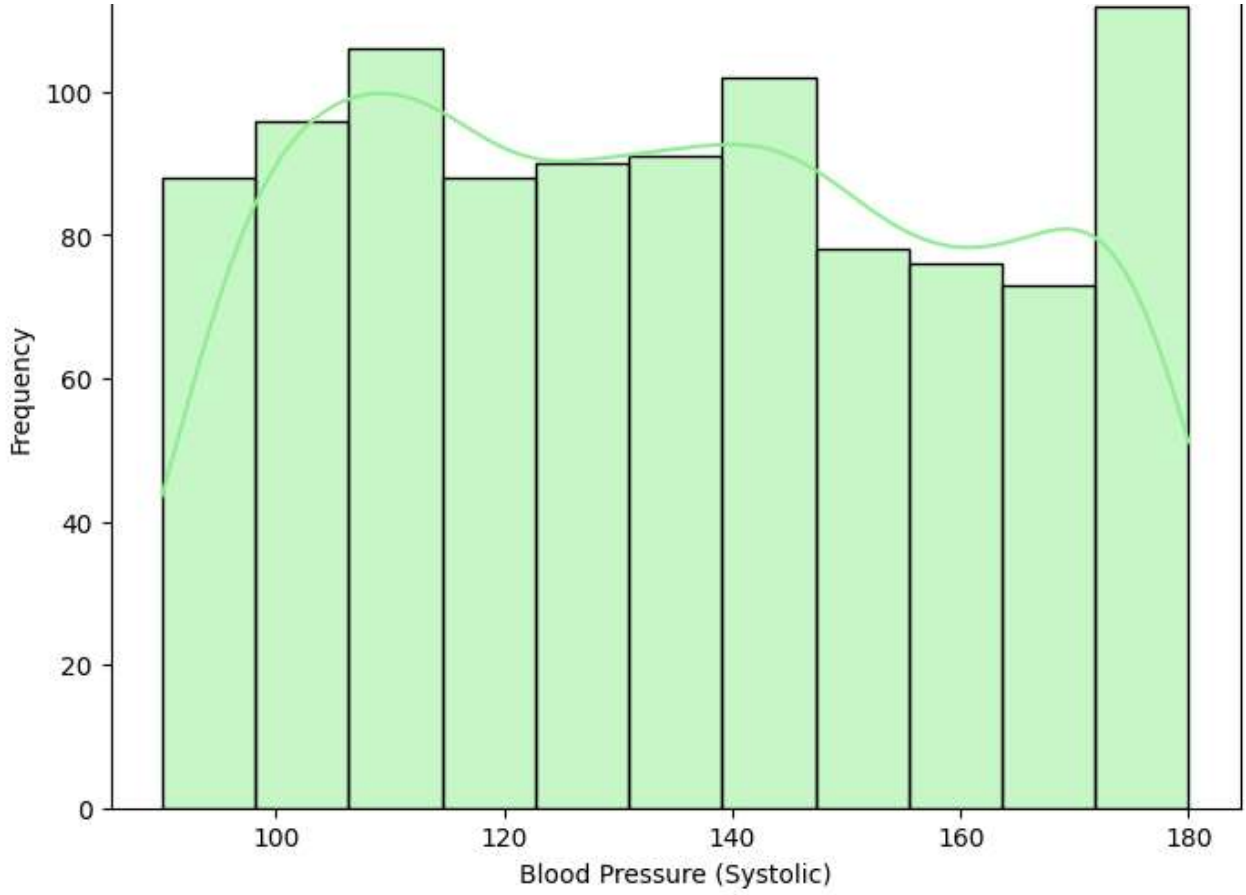
```
Skewness for numerical columns:
Age                          -0.029920
Temperature                  -0.060163
Blood Pressure (Systolic)     0.092493
Blood Pressure (Diastolic)    0.054590
Height                       -0.068798
Weight                       -0.048330
Needs Medical Attention      -1.113852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

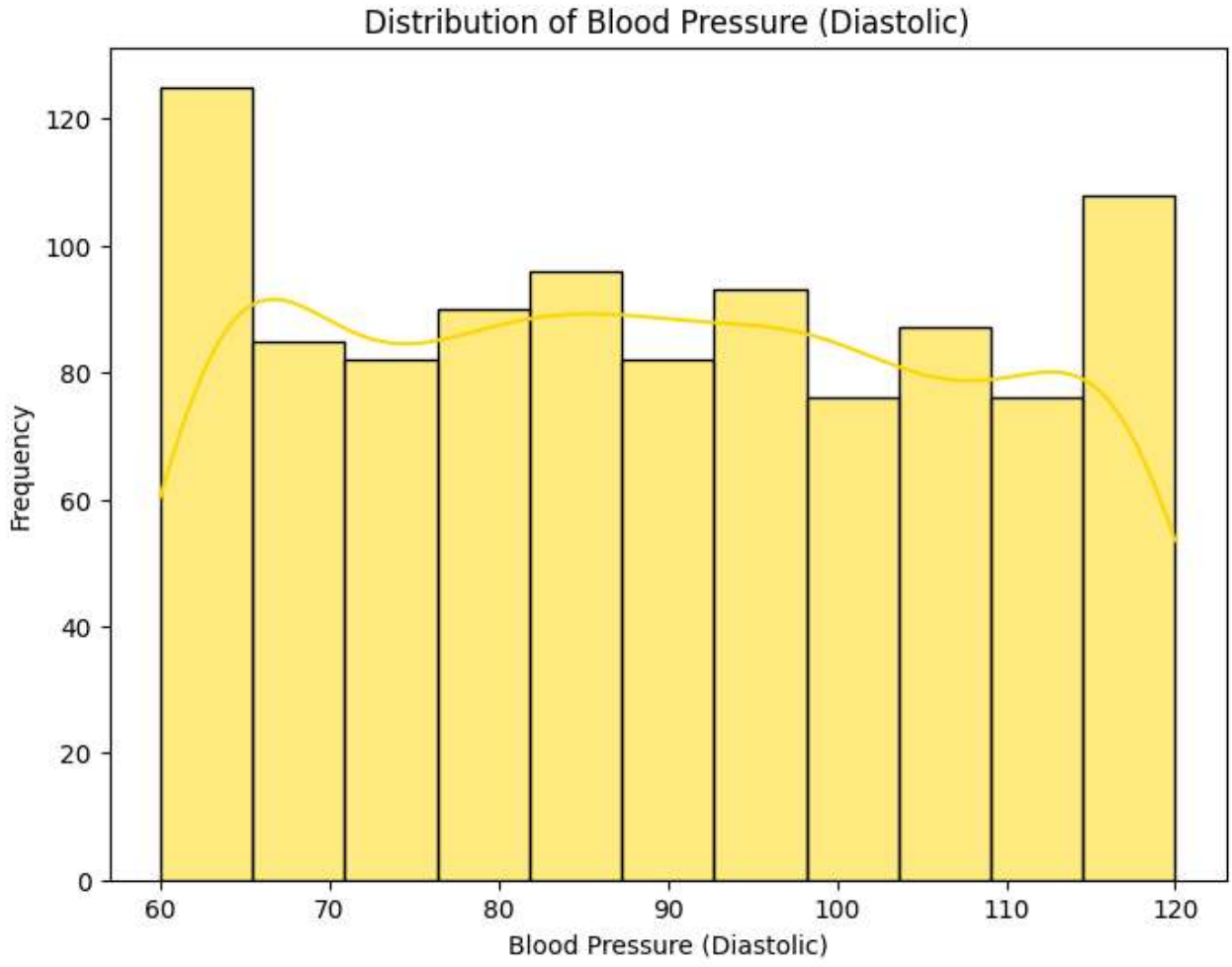
```
import matplotlib.pyplot as plt
import seaborn as sns

# Select numerical columns
numerical_columns = ['Age', 'Temperature', 'Blood Pressure (Systolic)', 'Blood Pressure (

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

# 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()
```

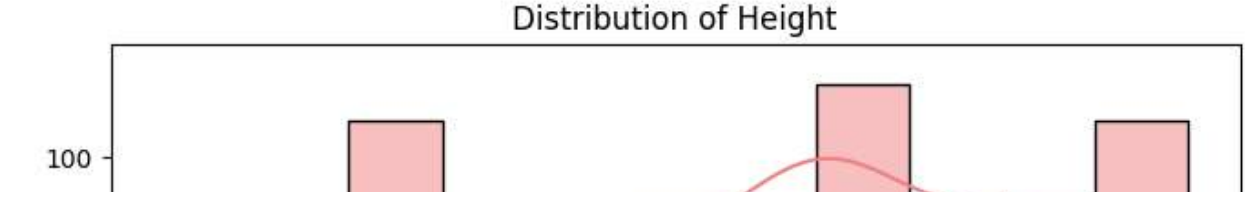

Distribution of Age
Frequency
Age
0
20
40
60
80
100

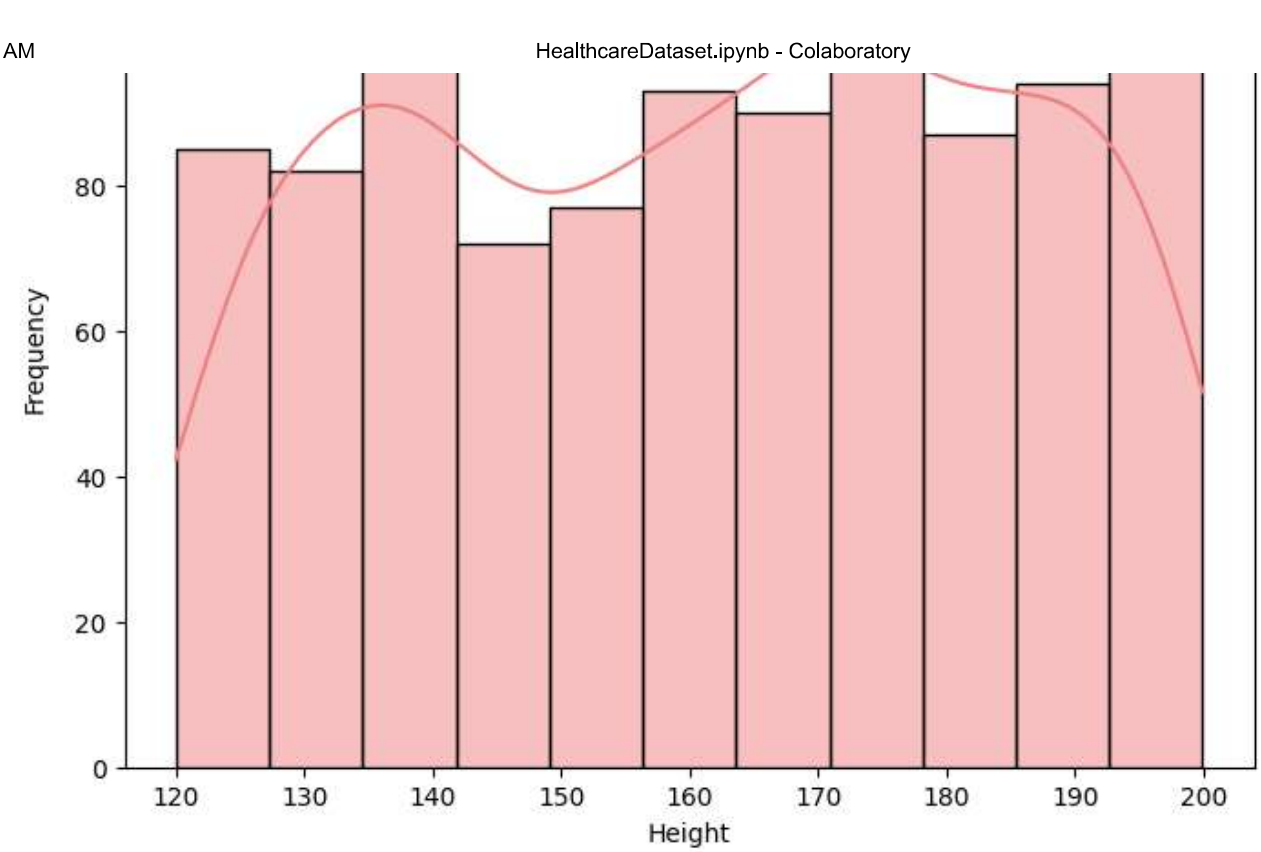

Distribution of Temperature
Frequency
Temperature
0
20
40
60
80
100
97.00
97.25
97.50
97.75
98.00
98.25
98.50
98.75
99.00

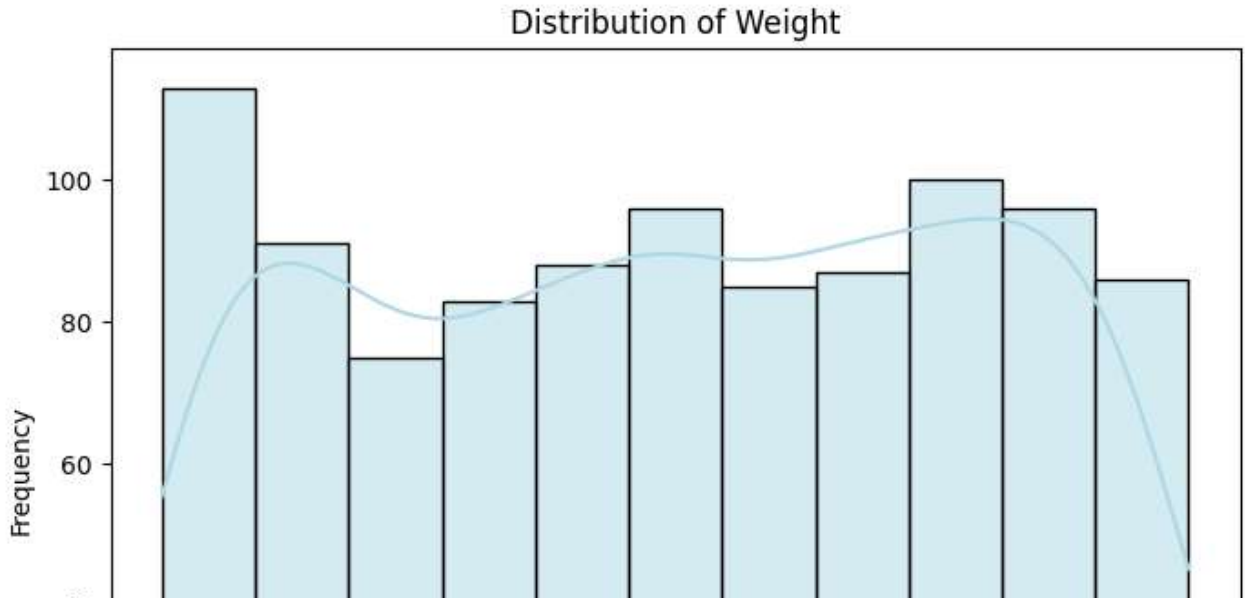

Distribution of Blood Pressure (Systolic)


Frequency
Blood Pressure (Systolic)


Distribution of Blood Pressure (Diastolic)
Frequency
Blood Pressure (Diastolic)


Distribution of Height


Frequency
0
20
40
60
80
120
130
140
150
160
170
180
190
200
Height


Distribution of Weight
Frequency
60
80
100

```
# 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)
```

```
221                     0
222                     1
227                     1
231                     0
247                     1
255                     1
269                     0
291                     0
334                     1
345                     1
356                     1
362                     1
399                     1
400                     1
403                     0
458                     1
481                     1
538                     1
568                     0
573                     1
577                     1
626                     1
649                     1
653                     1
654                     1
673                     1
704                     1
708                     1
713                     1
714                     1
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)	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	
...	...	...	...	...	...	...	...	...	

```
# 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 (Diasto
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	
...	...	...	...	...	...	...	...	...	

```
# 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, '

# 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)	Bloo 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
...	...	...	...	...	...	...	...	...	.

```
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)	Bloo 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

... ...

```
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
```