In [19]:

```python
import pandas as pd

# Sample dataset for testing
data = {
    'gender': ["Male", "Female", "Male", "Female"],
    'age': [23, 45, 36, 50],
    'blood_pressure': [120, 140, 130, 135]
}

# Create DataFrame
df = pd.DataFrame(data)

# Check the columns to ensure 'age' exists
print(df.columns)

# Check the first few rows
print(df.head())

# Example of handling missing values (just for testing)
df['age'] = df['age'].fillna(df['age'].median())  # Fill missing age values with the median
df['blood_pressure'] = df['blood_pressure'].fillna(df['blood_pressure'].median())

# Example of gender imputation (if needed)
df['gender'] = df['gender'].fillna(df['gender'].mode()[0])  # Fill missing gender with mode

# Verify the DataFrame after imputation
print(df.head())
```

```
Index(['gender', 'age', 'blood_pressure'], dtype='object')
   gender  age  blood_pressure
0    Male   23             120
1  Female   45             140
2    Male   36             130
3  Female   50             135
   gender  age  blood_pressure
0    Male   23             120
1  Female   45             140
2    Male   36             130
3  Female   50             135
```

In [3]:

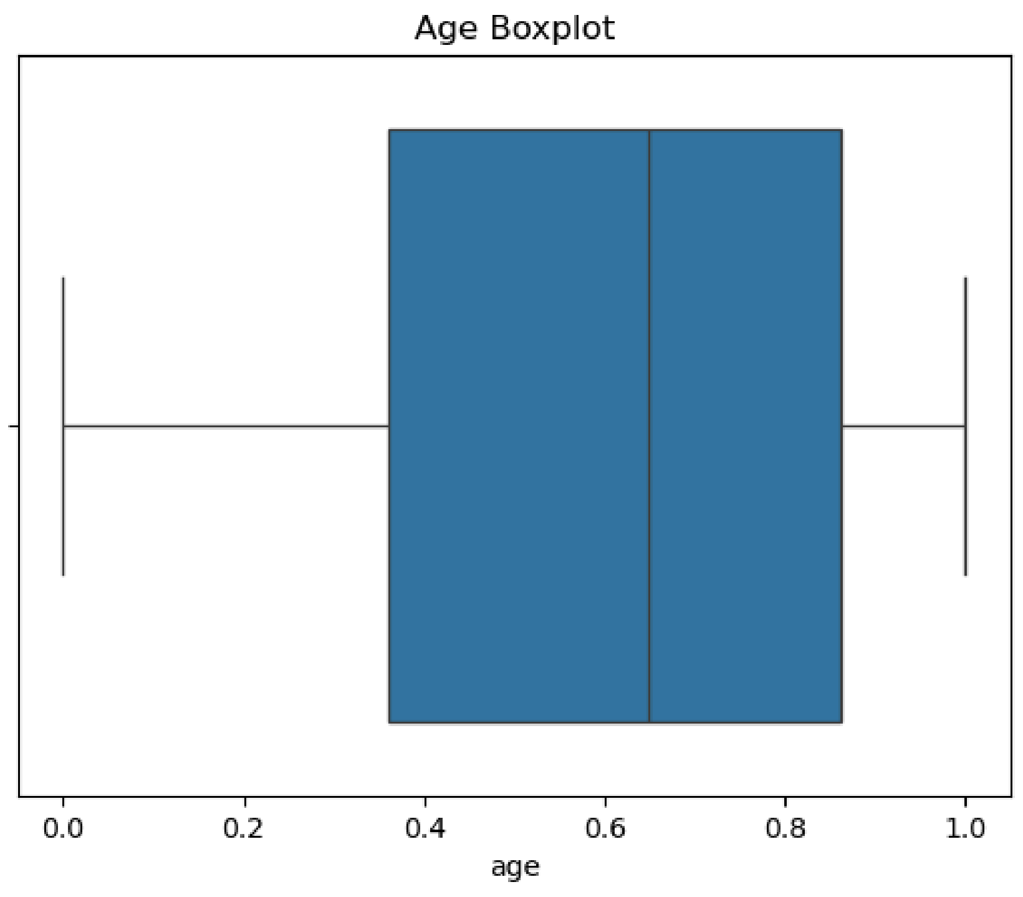
```python
# 3. Detect and Handle Duplicates

# Identify duplicates
duplicates = df.duplicated().sum()
print(f'Number of duplicate rows: {duplicates}')

# Remove duplicate rows
df = df.drop_duplicates()

# After removing duplicates, check if any remain
print(f'Number of duplicate rows after cleaning: {df.duplicated().sum()}')
```

```
Number of duplicate rows: 5
Number of duplicate rows after cleaning: 0
```

In [12]:

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler

# Sample dataset for demonstration
data = {
    'gender': ["Male", "Female", "Male", "Female"],
    'age': [23, 45, 36, 50],
    'blood_pressure': [120, 140, 130, 135]
}

# Load data into a DataFrame
df = pd.DataFrame(data)

# Verify the columns to ensure correct names
print(df.columns)  # Verify column names here

# 5. Standardize and Normalize Data

# Convert categorical variables to numerical representations (e.g., 'gender' -> 0 or 1)
df['gender'] = df['gender'].map({'Male': 0, 'Female': 1})  # Mapping 'Male' to 0, 'Female' to 1

# Scale numerical variables (e.g., 'age', 'blood_pressure')
scaler = MinMaxScaler()
df[['age', 'blood_pressure']] = scaler.fit_transform(df[['age', 'blood_pressure']])

# After scaling, you can verify the data
print(df.head())

# 6. Detect and Handle Outliers (Using Boxplot)

# Check the column names and then use the correct one
sns.boxplot(x=df['age'])  # Ensure 'age' exists, otherwise update with the correct column name
plt.title('Age Boxplot')
plt.show()
```

```
Index(['gender', 'age', 'blood_pressure'], dtype='object')
   gender       age  blood_pressure
0       0  0.000000            0.00
1       1  0.814815            1.00
2       0  0.481481            0.50
3       1  1.000000            0.75
```



In [13]:

```python
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Example data for gender, age, and blood pressure
data = {
    'gender': ["Male", "Female", "Male", "Female"],
    'age': [23, 45, 36, 50],
    'blood_pressure': [120, 140, 130, 135]
}

# Load data into a DataFrame
df = pd.DataFrame(data)

# 5. Standardize and Normalize Data

# Convert categorical variables to numerical representations (e.g., 'gender' -> 0 or 1)
df['gender'] = df['gender'].map({'Male': 0, 'Female': 1})  # Assuming 'Male' = 0 and 'Female' = 1

# Scale numerical variables (e.g., 'age', 'blood_pressure') using Min-Max Scaling
scaler = MinMaxScaler()
df[['age', 'blood_pressure']] = scaler.fit_transform(df[['age', 'blood_pressure']])

# After scaling, you can verify the data
print(df.head())

# 6. Data Validation

# Ensure no missing values
assert df.isna().sum().sum() == 0, "There are still missing values in the dataset!"

# Ensure no duplicates remain
assert df.duplicated().sum() == 0, "There are still duplicate rows in the dataset!"

# Check data types for correctness
print(df.dtypes)

# 7. Final Data Export

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

print("Data cleaning completed and saved as 'cleaned_healthcare_data.csv'.")
```

```
   gender       age  blood_pressure
0       0  0.000000            0.00
1       1  0.814815            1.00
2       0  0.481481            0.50
3       1  1.000000            0.75
gender              int64
age               float64
blood_pressure    float64
dtype: object
Data cleaning completed and saved as 'cleaned_healthcare_data.csv'.
```

In [16]:

```python
import pandas as pd

# Sample dataset for testing
data = {
    'gender': ["Male", "Female", "Male", "Female"],
    'age': [23, 45, 36, 50],
    'blood_pressure': [120, 140, 130, 135]
}

# Create DataFrame
df = pd.DataFrame(data)

# Check the columns to ensure 'age' exists
print(df.columns)

# Check the first few rows
print(df.head())

# Example of handling missing values (just for testing)
df['age'] = df['age'].fillna(df['age'].median())  # Fill missing age values with the median
df['blood_pressure'] = df['blood_pressure'].fillna(df['blood_pressure'].median())

# Example of gender imputation (if needed)
df['gender'] = df['gender'].fillna(df['gender'].mode()[0])  # Fill missing gender with mode

# Verify the DataFrame after imputation
print(df.head())
```

```
Index(['gender', 'age', 'blood_pressure'], dtype='object')
   gender  age  blood_pressure
0    Male   23             120
1  Female   45             140
2    Male   36             130
3  Female   50             135
   gender  age  blood_pressure
0    Male   23             120
1  Female   45             140
```

```
2    Male   36             130
3  Female   50             135
```

In [ ]:

```

```