US Demographics & Cardiovascular Diseases

The dataset provides a wealth of information on health parameters and demographics of US residents, making it ideal for exploring cardiovascular diseases. Through statistical analysis, the project aims to uncover insights transcending disciplinary boundaries. Methods like confidence interval estimation allow for robust inference on demographic parameters. Hypothesis testing helps to statistically verify claims on the demographic parameters, as well as identifies significant associations between population characteristics and heart diseases: aiding evidence-based strategies for public health and equitable outcomes.

Importing the Necessary Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as sc
import imblearn
from scipy.stats import norm
import math
import scipy.stats as stats
from scipy.stats import chi2_contingency

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/undersampled-data/Undersampled_Data.csv
/kaggle/input/cardiovascular-diseases-risk-prediction-dataset/CVD_cleaned.csv
```

Loading the Dataset

In [2]:
```
dfo = pd.read_csv('/kaggle/input/cardiovascular-diseases-risk-prediction-dataset/CVD_cleaned.cs
dfo['index'] = dfo.index
id_col = dfo['index']
dfo.drop_duplicates()
dfo.insert(loc = 0, column = 'ID', value = id_col)
dfo.drop('index', axis=1, inplace=True)
dfo.head()
```

Out[2]:

	ID	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabet
0	0	Poor	Within the past 2 years	No	No	No	No	No	I
1	1	Very Good	Within the past year	No	Yes	No	No	No	Y
2	2	Very Good	Within the past year	Yes	No	No	No	No	Y
3	3	Poor	Within the past year	Yes	Yes	No	No	No	Y
4	4	Good	Within the past year	No	No	No	No	No	I

In [3]:
```
dfo.describe()
```

Out[3]:

	ID	Height_(cm)	Weight_(kg)	BMI	Alcohol_Consumption	Fruit_Consum
count	308854.000000	308854.000000	308854.000000	308854.000000	308854.000000	308854.00
mean	154426.500000	170.615249	83.588655	28.626211	5.096366	29.83
std	89158.614358	10.658026	21.343210	6.522323	8.199763	24.87
min	0.000000	91.000000	24.950000	12.020000	0.000000	0.00
25%	77213.250000	163.000000	68.040000	24.210000	0.000000	12.00
50%	154426.500000	170.000000	81.650000	27.440000	1.000000	30.00
75%	231639.750000	178.000000	95.250000	31.850000	6.000000	30.00
max	308853.000000	241.000000	293.020000	99.330000	30.000000	120.00

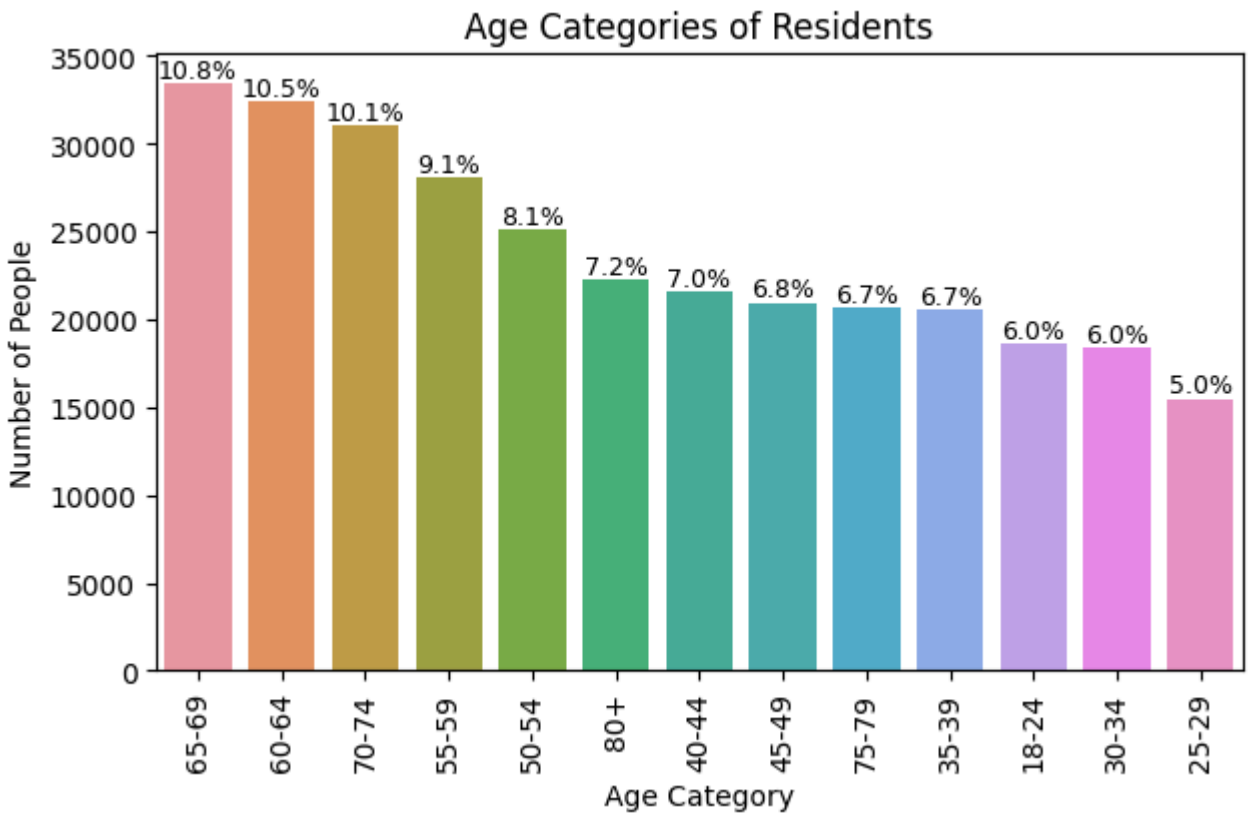
1. Statistical Inference on Patient Demographic Data

1.1 Age & Gender

- Considering the age range of residents is crucial for studying health trends. Elderly people are more prone to certain diseases than the young.
- We look at how many people are in different age groups in the dataset. Residents in the dataset are between 18 and 85 years old, with a median age of 45 years.

- This information helps healthcare providers and insurance companies design age-specific health plans and policies.
- Dividing the ages from 18 to 80+ into 13 parts is a useful way to organize the data for analysis.

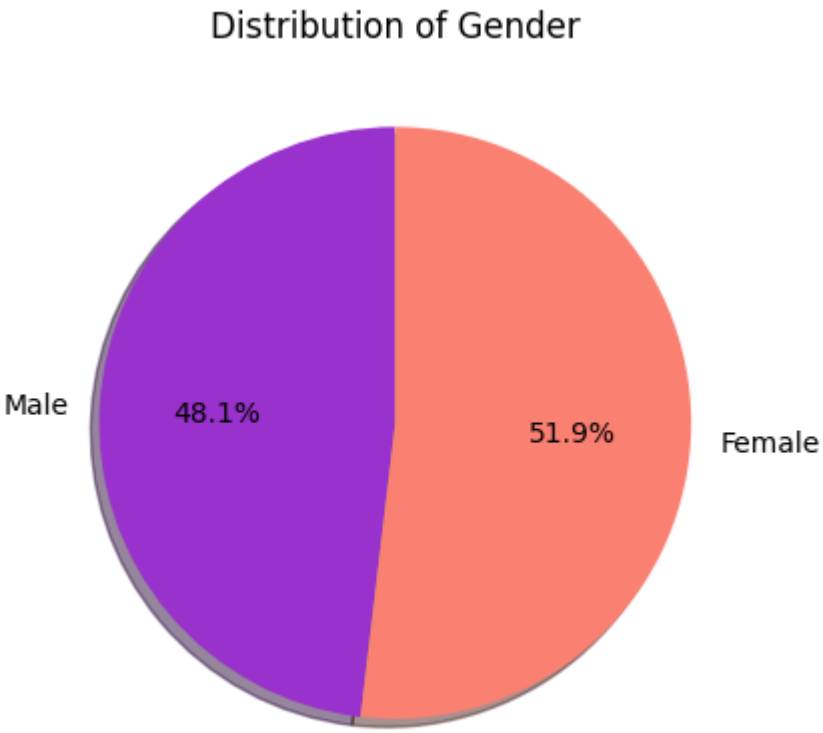
In [4]:
```
plt.figure(figsize=(7,4))
ages = dfo["Age_Category"]
ax = sns.countplot(data = dfo, x="Age_Category", order=ages.value_counts().index)
plt.xticks(rotation=90)
plt.title("Age Categories of Residents")
plt.xlabel("Age Category")
plt.ylabel("Number of People")
patches = ax.patches
for i in range(len(patches)):
    offset = ages.value_counts().max() * 0.01
    per_values = list(ages.value_counts())[i]/ages.value_counts().sum()
    x = patches[i].get_x() + patches[i].get_width()/2
    y = patches[i].get_height()+ offset
    ax.annotate('{:.1f}%'.format(per_values*100), (x, y), ha='center', fontsize=9)

plt.show()
```



- It is important to consider gender as a factor in the healthcare system.
- Men, women, and others may have different health needs and risks.
- Our dataset consists of two genders viz. Male and Female, and the proportion is given in the following figure.

In [5]:
```
males = dfo.loc[dfo["Sex"]=="Male", "Sex"].value_counts()
males = int(males.iloc[0])
females = dfo.loc[dfo["Sex"]=="Female", "Sex"].value_counts()
females = int(females.iloc[0])
```

```
fig, ax = plt.subplots()
labels = ["Male", "Female"]
sizes = [males, females]
ax.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=True, startangle=90, colors=["Darkorchid

plt.title("Distribution of Gender")
plt.show()
```



1.2 Analysis of Height Data

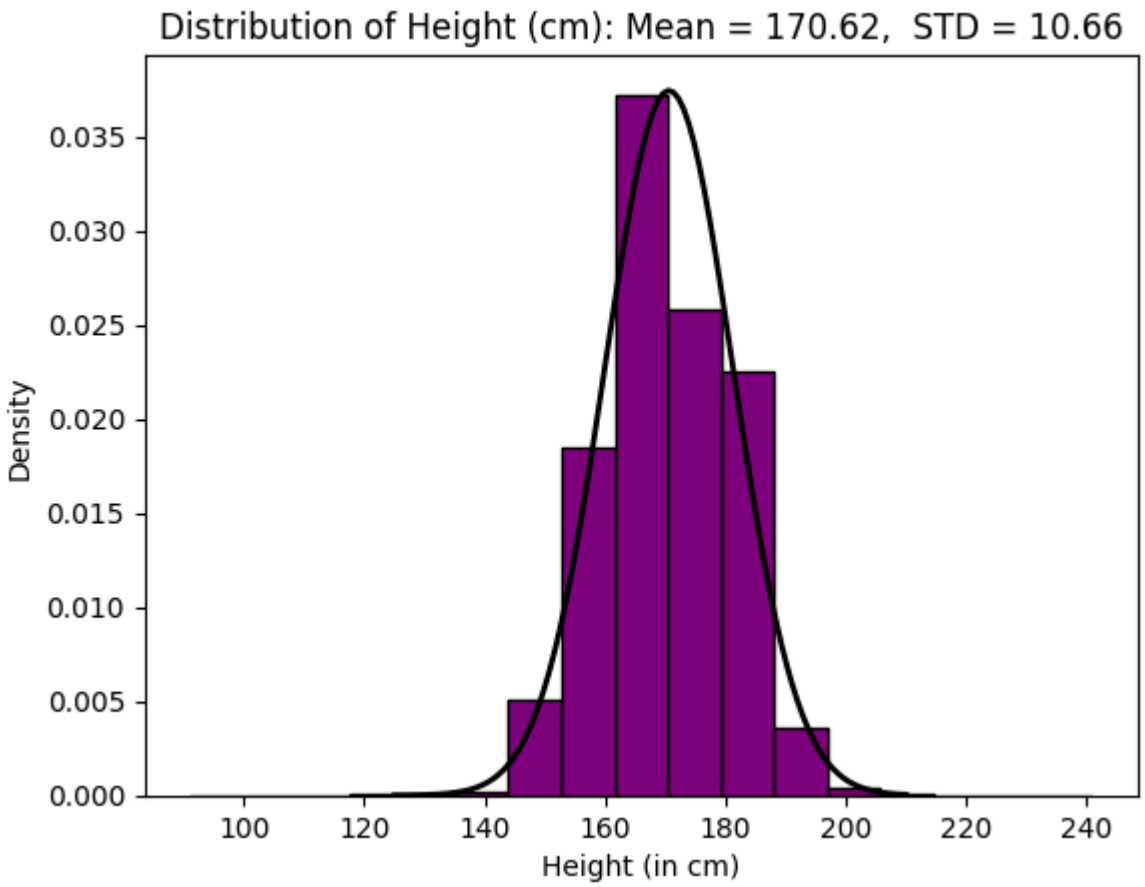
1.2.1 Distribution of Height (cm) of all Patients

Height of the US residents is approximately normally distributed with mean $170.62\ cm$ and variance $113.59\ cm^2$.

In [6]:
```
from scipy.stats import norm

data = dfo['Height_(cm)'].tolist()

# Fit a normal distribution to the data:
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=17, density=True, ec='k', color='purple')


# Plot the PDF.

x = np.linspace(125, 210, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
title = "Distribution of Height (cm): Mean = %.2f,  STD = %.2f" % (mu, std)
```

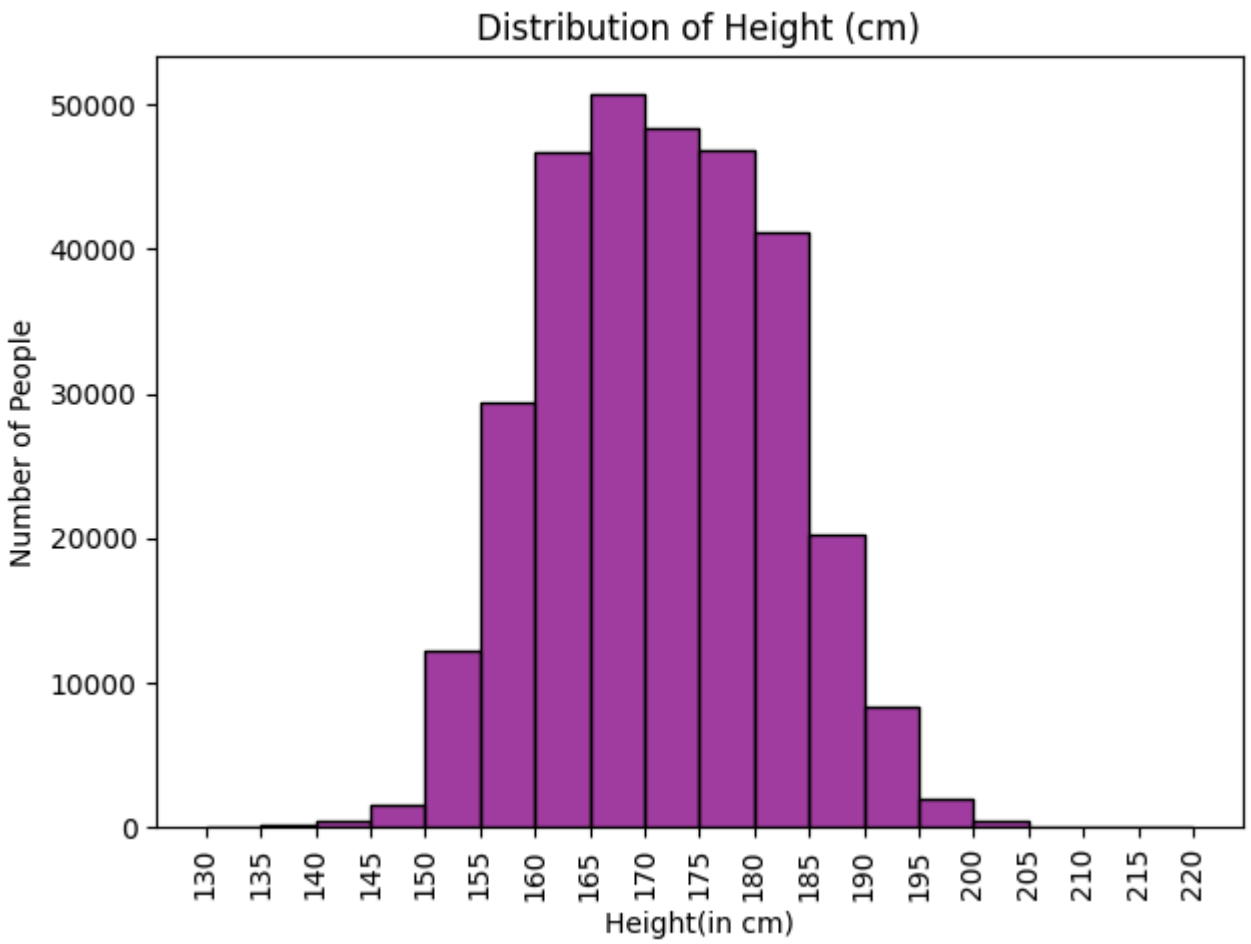
```
plt.title(title)
plt.xlabel("Height (in cm)")
plt.ylabel("Density")
```

Out[6]: Text(0, 0.5, 'Density')



In [7]:
```
plt.figure(figsize=(7,5))
plt.title("Distribution of Height (cm)")
ax = sns.histplot(dfo["Height_(cm)"], bins=[x for x in range(130,221,5)], color='purple')
plt.ylabel("Number of People")
plt.xlabel("Height(in cm)")
plt.xticks([x for x in range(130, 221, 5)], rotation=90)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na o
ption is deprecated and will be removed in a future version. Convert inf values to NaN before op
erating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Confidence Interval Estimation for Mean Height of the Population

t-values for 95% and 99% CIs with n-1 degrees of freedom (where n = number of samples)

We need the value of $t_{\frac{\alpha}{2},n-1}$, where for 95% CI, $\alpha = 0.05$.

Note that to the t.ppf function, we pass $q = 1 - \frac{\alpha}{2}$, that is $1 - 0.025 = 0.975$.

In [8]:
```
import scipy.stats as stats
n = dfo.shape[0]
# Note that the value of parameter q is 1 - (alpha/2). For 95% CI, we have alpha = 0.05
t_95 = stats.t.ppf(q = 0.975, df=n-1)
t_99 = stats.t.ppf(q = 0.995, df=n-1)
print("t_95 = ", t_95)
print("t_99 = ", t_99)
```

```
t_95 =  1.959971665476839
t_99 =  2.575845222371783
```

Let $X_1, X_2, \ldots, X_n$ are normally distributed with unknown mean μ and variance σ^2, then a $(1-\alpha)100\%$ confidence interval for the population mean μ is:

$$\left(\overline{X} - t_{\alpha/2,n-1}\left(\frac{S}{\sqrt{n}}\right),\ \overline{X} + t_{\alpha/2,n-1}\left(\frac{S}{\sqrt{n}}\right)\right)$$

In [9]:
```
heights = dfo["Height_(cm)"].tolist()
n = len(heights)
```

```
sample_mean = np.mean(heights)
sample_std = np.std(heights)
print("Point estimate for population mean height (Sample mean): ", round(sample_mean, 3))

#Lower bound for the CI of population mean (95% confidence)
lb_pop_mean_95 = sample_mean - (t_95*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (95% confidence)
ub_pop_mean_95 = sample_mean + (t_95*(sample_std/math.sqrt(n)))

#Lower bound for the CI of population mean (99% confidence)
lb_pop_mean_99 = sample_mean - (t_99*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (99% confidence)
ub_pop_mean_99 = sample_mean + (t_99*(sample_std/math.sqrt(n)))

print("The 95% Confidence Interval for the mean height of the population: ({},{})".format(round
print("The 99% Confidence Interval for the mean height of the population: ({}, {})".format(roun
```

```
Point estimate for population mean height (Sample mean):  170.615
The 95% Confidence Interval for the mean height of the population: (170.578,170.653)
The 99% Confidence Interval for the mean height of the population: (170.566, 170.665)
```

Confidence Interval for Population Variance & STD of Height

$a = \chi^2_{1-\frac{\alpha}{2},n-1}$

$b = \chi^2_{\frac{\alpha}{2},n-1}$

Then a $(1-\alpha)\times 100\%$ CI for the population variance is:

$(\frac{(n-1)S^2}{b}, \frac{(n-1)S^2}{a})$

Note that in the **stats.chi2.ppf()** function, we need to feed $\alpha/2$ for calculating a and $1-\alpha/2$ for calculating b.

In [10]:
```
n = dfo.shape[0]
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
```

In [11]:
```
print("Chi^2 value for 95% CI (alpha = 0.05) and (n-1) degrees of freedom: ", a_95)
print("Chi^2 value for 95% CI (alpha = 0.05) and (n-1) degrees of freedom: ", b_95)
print("Chi^2 value for 99% CI (alpha = 0.05) and (n-1) degrees of freedom: ", a_99)
print("Chi^2 value for 99% CI (alpha = 0.05) and (n-1) degrees of freedom: ", b_99)
```

```
Chi^2 value for 95% CI (alpha = 0.05) and (n-1) degrees of freedom:  307314.47570410214
Chi^2 value for 95% CI (alpha = 0.05) and (n-1) degrees of freedom:  310395.3129058949
Chi^2 value for 99% CI (alpha = 0.05) and (n-1) degrees of freedom:  306832.3023720106
Chi^2 value for 99% CI (alpha = 0.05) and (n-1) degrees of freedom:  310881.2108182607
```

In [12]:
```
# Population Variance of Heights

heights = dfo["Height_(cm)"].tolist()
n = len(heights)
N = (n-1)/n

sample_var = np.var(heights)
print("Point estimate for the population variance of height: ", round(N*sample_var,3))
```

```
#Lower bound for the CI of population variance (95% confidence)
lb_pop_var_95 = ((n-1)*sample_var)/b_95
#Upper bound for the CI of population variance (95% confidence)
ub_pop_var_95 = ((n-1)*sample_var)/a_95

#Lower bound for the CI of population variance (99% confidence)
lb_pop_var_99 = ((n-1)*sample_var)/b_99
#Upper bound for the CI of population variance (99% confidence)
ub_pop_var_99 = ((n-1)*sample_var)/a_99

print("The 95% Confidence Interval for the population variance of height: ({}, {})".format(roun
print("The 99% Confidence Interval for the population variance of height: ({}, {})".format(roun
```

```
Point estimate for the population variance of height:  113.593
The 95% Confidence Interval for the population variance of height: (113.029, 114.162)
The 99% Confidence Interval for the population variance of height: (112.852, 114.341)
```

In [13]:
```
# Population Standard Deviation of Heights

heights = dfo["Height_(cm)"].tolist()
n = len(heights)
N = (n-1)/n

sample_std = np.std(heights)
print("Point estimate for the standard deviation: ", round(math.sqrt(N)*sample_std,3))

#Lower bound for the CI of population STD (95% confidence)
lb_pop_std_95 = math.sqrt((n-1)/b_95)*sample_std
#Upper bound for the CI of population STD (95% confidence)
ub_pop_std_95 = math.sqrt((n-1)/a_95)*sample_std

#Lower bound for the CI of population STD (99% confidence)
lb_pop_std_99 = math.sqrt((n-1)/b_99)*sample_std
#Upper bound for the CI of population STD (99% confidence)
ub_pop_std_99 = math.sqrt((n-1)/a_99)*sample_std

print("The 95% Confidence Interval for the population STD of height: ({}, {})".format(round(lb_
print("The 99% Confidence Interval for the population STD of height: ({}, {})".format(round(lb_
```

```
Point estimate for the standard deviation:  10.658
The 95% Confidence Interval for the population STD of height: (10.631, 10.685)
The 99% Confidence Interval for the population STD of height: (10.623, 10.693)
```

1.2.2 Height Distribution by Gender: Male Residents

In [14]:
```
df_male = dfo.loc[dfo["Sex"]=="Male", :]
df_male.head()
```

Out[14]:

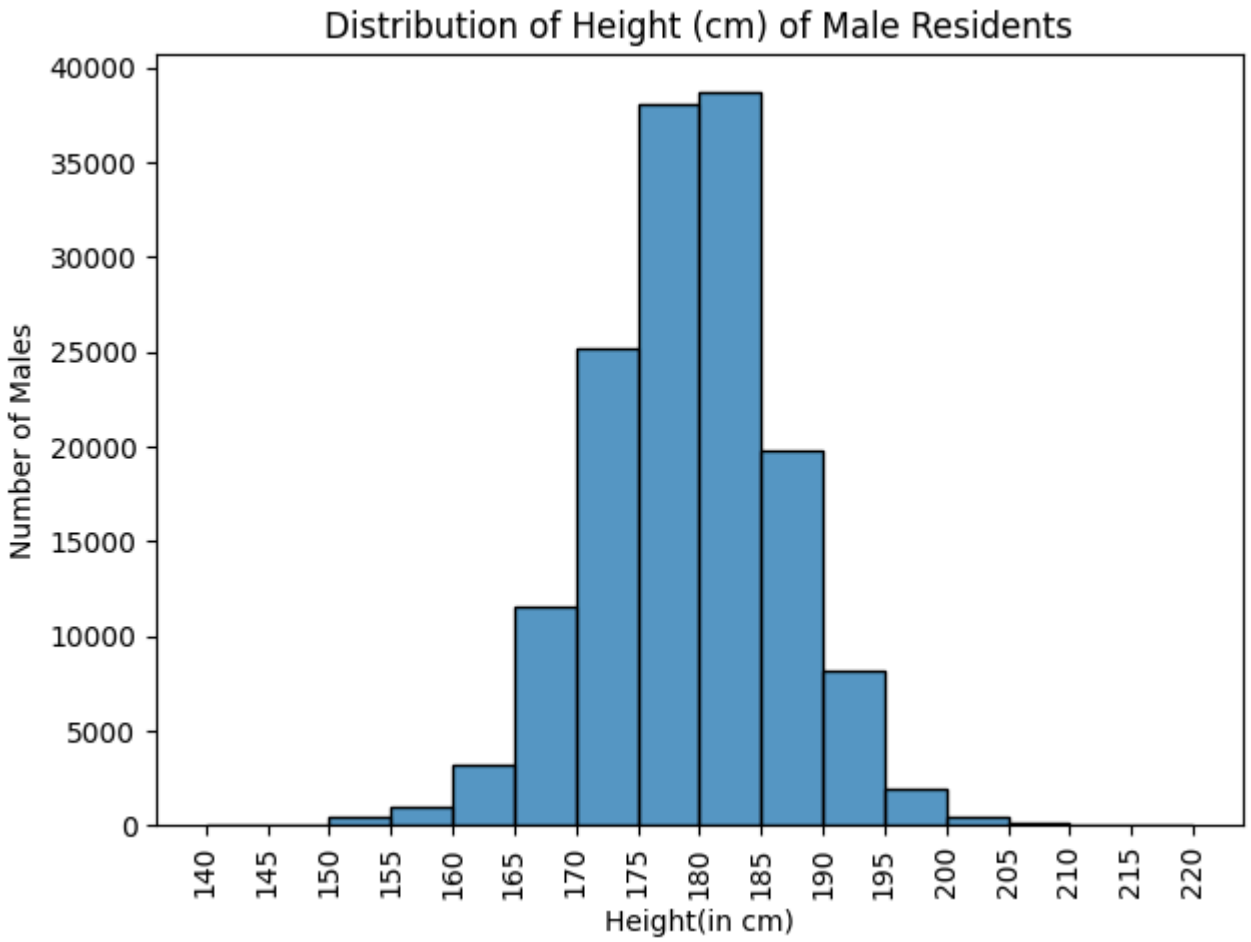
	ID	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabe
3	3	Poor	Within the past year	Yes	Yes	No	No	No	
4	4	Good	Within the past year	No	No	No	No	No	
5	5	Good	Within the past year	No	No	No	No	Yes	
6	6	Fair	Within the past year	Yes	Yes	No	No	No	
11	11	Fair	Within the past year	No	Yes	Yes	No	No	

In [15]:
```
print("Number of male residents:", df_male.shape[0])
```

```
Number of male residents: 148658
```

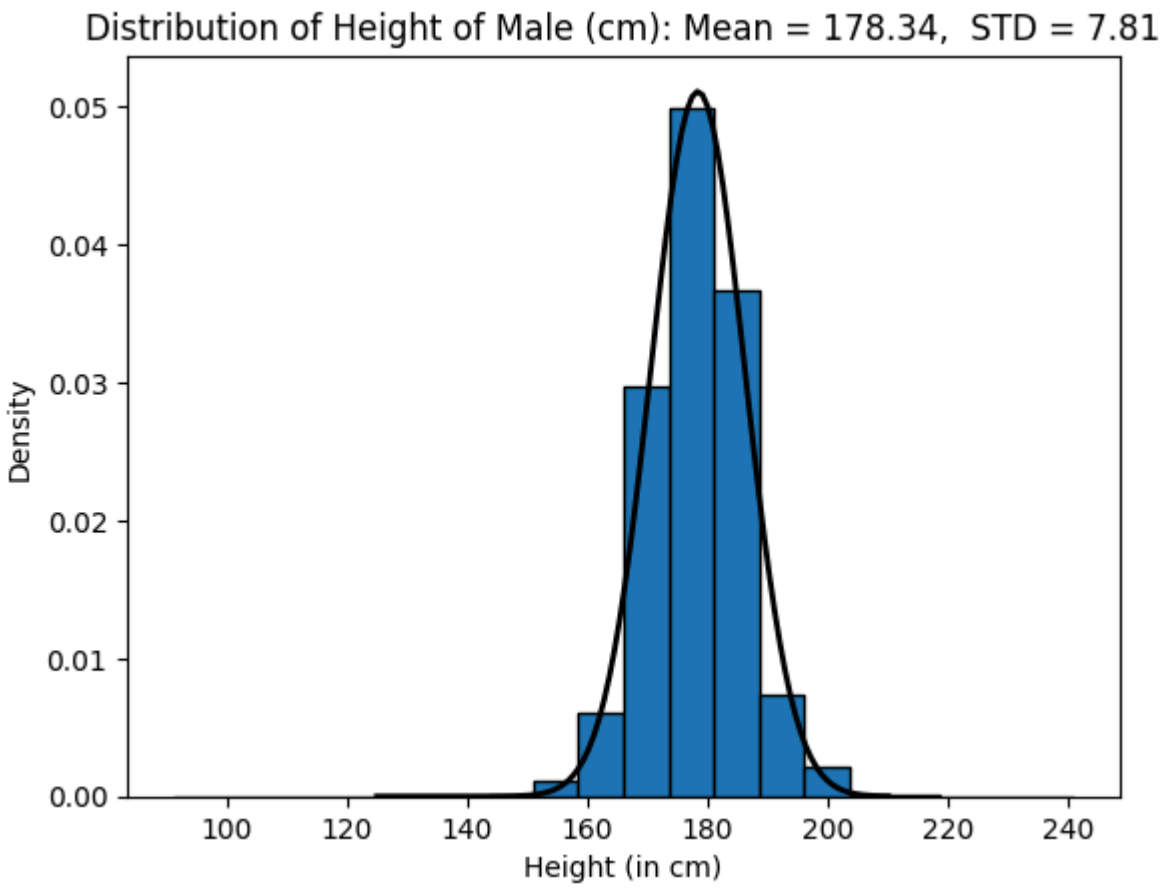
In [16]:
```
#Height of Male Residents

plt.figure(figsize=(7,5))
plt.title("Distribution of Height (cm) of Male Residents")
ax = sns.histplot(df_male["Height_(cm)"], bins=[x for x in range(140,221,5)])
plt.ylabel("Number of Males")
plt.xlabel("Height(in cm)")
plt.xticks([x for x in range(140, 221, 5)], rotation=90)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [17]:
```
from scipy.stats import norm

data = df_male['Height_(cm)'].tolist()

# Fit a normal distribution to the data:
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=20, density=True, ec='k')


# Plot the PDF.

x = np.linspace(125, 210, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
title = "Distribution of Height of Male (cm): Mean = %.2f,  STD = %.2f" % (mu, std)
plt.title(title)
plt.xlabel("Height (in cm)")
plt.ylabel("Density")
```

Out[17]:
```
Text(0, 0.5, 'Density')
```



In [18]:
```
# Confidence Interval Estimation for Mean Height of Male Population

male_heights = df_male["Height_(cm)"].tolist()
n = len(male_heights)
t_95 = stats.t.ppf(q = 0.975, df=n-1)
t_99 = stats.t.ppf(q = 0.995, df=n-1)

sample_mean = np.mean(male_heights)
sample_std = np.std(male_heights)
print("Point estimate for the mean population height of the male patients: ", round(sample_mean

#Lower bound for the CI of population mean (95% confidence)
lb_pop_mean_95 = sample_mean - (t_95*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (95% confidence)
ub_pop_mean_95 = sample_mean + (t_95*(sample_std/math.sqrt(n)))

#Lower bound for the CI of population mean (99% confidence)
lb_pop_mean_99 = sample_mean - (t_99*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (99% confidence)
ub_pop_mean_99 = sample_mean + (t_99*(sample_std/math.sqrt(n)))

print("The 95% Confidence Interval for the mean height of the Male population: ({},{})".format(
print("The 99% Confidence Interval for the mean height of the Male population: ({}, {})".format
```

```
Point estimate for the mean population height of the male patients:  178.34
The 95% Confidence Interval for the mean height of the Male population: (178.3,178.379)
The 99% Confidence Interval for the mean height of the Male population: (178.288, 178.392)
```

In [19]:
```
# Confidence Interval Estimation for STD of Height of Male Population

male_heights = df_male["Height_(cm)"].tolist()
n = len(male_heights)
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
```

```
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
N = (n-1)/n

sample_std = np.std(male_heights)
print("Point estimate for the STD of height of the population of male patients: ", round(math.s

#Lower bound for the CI of population STD (95% confidence)
lb_pop_std_95 = math.sqrt((n-1)/b_95)*sample_std
#Upper bound for the CI of population STD (95% confidence)
ub_pop_std_95 = math.sqrt((n-1)/a_95)*sample_std

#Lower bound for the CI of population STD (99% confidence)
lb_pop_std_99 = math.sqrt((n-1)/b_99)*sample_std
#Upper bound for the CI of population STD (99% confidence)
ub_pop_std_99 = math.sqrt((n-1)/a_99)*sample_std

print("The 95% Confidence Interval for population STD of height of male patients: ({}, {})".for
print("The 99% Confidence Interval for population STD of height of male patients: ({}, {})".for
```

```
Point estimate for the STD of height of the population of male patients:  7.808
The 95% Confidence Interval for population STD of height of male patients: (7.78, 7.836)
The 99% Confidence Interval for population STD of height of male patients: (7.771, 7.845)
```

In [20]:

```
# Population Variance of Heights of Male Patients

male_heights = df_male["Height_(cm)"].tolist()
n = len(male_heights)
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
N = (n-1)/n


sample_var = np.var(male_heights)
print("Point estimate for the population variance of height of male patients: ", round(N*sample

#Lower bound for the CI of population variance (95% confidence)
lb_pop_var_95 = ((n-1)*sample_var)/b_95
#Upper bound for the CI of population variance (95% confidence)
ub_pop_var_95 = ((n-1)*sample_var)/a_95

#Lower bound for the CI of population variance (99% confidence)
lb_pop_var_99 = ((n-1)*sample_var)/b_99
#Upper bound for the CI of population variance (99% confidence)
ub_pop_var_99 = ((n-1)*sample_var)/a_99

print("The 95% Confidence Interval for the population variance of height of male patients: ({},
print("The 99% Confidence Interval for the population variance of height of male patients: ({},
```

```
Point estimate for the population variance of height of male patients:  60.966
The 95% Confidence Interval for the population variance of height of male patients: (60.53, 61.4
07)
The 99% Confidence Interval for the population variance of height of male patients: (60.394, 61.
546)
```

1.2.3 Height Distribution by Gender: Female Residents

In [21]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
df_female.head()
```

Out[21]:

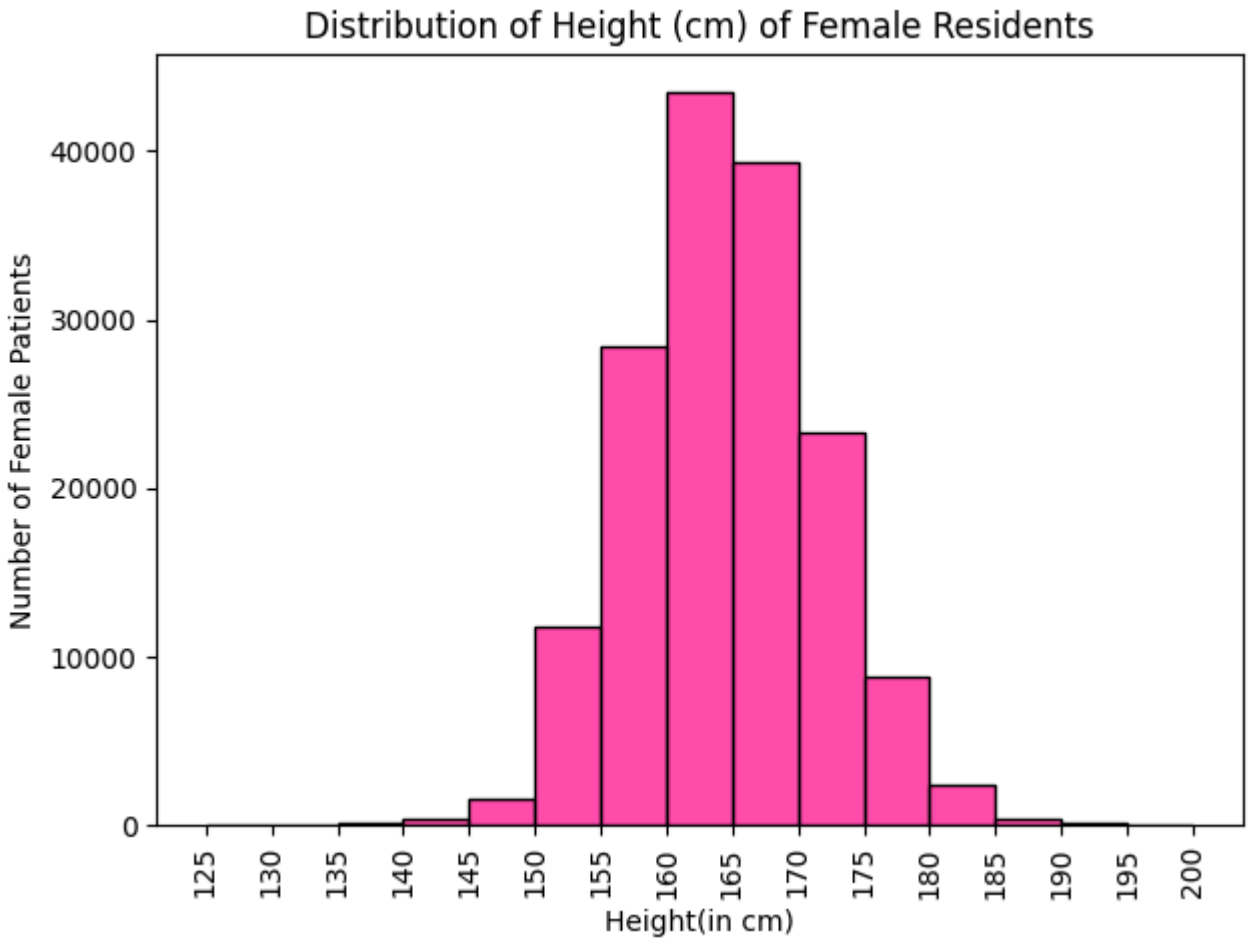
	ID	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	Diabet
0	0	Poor	Within the past 2 years	No	No	No	No	No	I
1	1	Very Good	Within the past year	No	Yes	No	No	No	Y
2	2	Very Good	Within the past year	Yes	No	No	No	No	Y
7	7	Good	Within the past year	Yes	No	No	No	No	I
8	8	Fair	Within the past year	No	No	No	No	Yes	I

In [22]:
```
print("Number of female residents:", df_female.shape[0])
```

```
Number of female residents: 160196
```

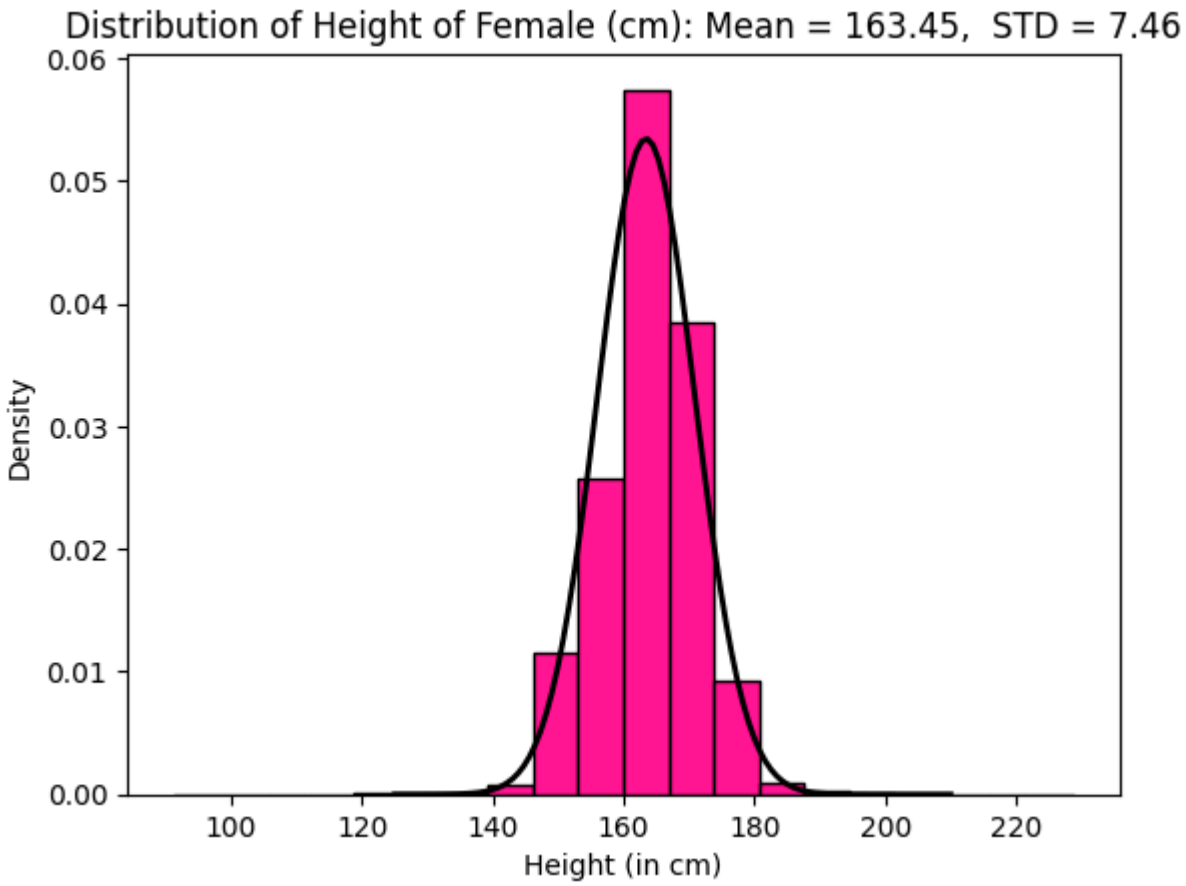
In [23]:
```
#Height of Female Patients

plt.figure(figsize=(7,5))
plt.title("Distribution of Height (cm) of Female Residents")
ax = sns.histplot(df_female["Height_(cm)"], bins=[x for x in range(125,201,5)], color='deeppink
plt.ylabel("Number of Female Patients")
plt.xlabel("Height(in cm)")
plt.xticks([x for x in range(125, 201, 5)], rotation=90)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na o
ption is deprecated and will be removed in a future version. Convert inf values to NaN before op
erating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [24]:
```
from scipy.stats import norm

data = df_female['Height_(cm)'].tolist()

# Fit a normal distribution to the data:
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=20, density=True, ec='k', color='deeppink')


# Plot the PDF.

x = np.linspace(125, 210, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
title = "Distribution of Height of Female (cm): Mean = %.2f,  STD = %.2f" % (mu, std)
plt.title(title)
plt.xlabel("Height (in cm)")
plt.ylabel("Density")
```

Out[24]:
```
Text(0, 0.5, 'Density')
```



In [25]:
```
# Confidence Interval Estimation for Mean Height of Female Population

female_heights = df_female["Height_(cm)"].tolist()
n = len(female_heights)

t_95 = stats.t.ppf(q = 0.975, df=n-1)
t_99 = stats.t.ppf(q = 0.995, df=n-1)

sample_mean = np.mean(female_heights)
sample_std = np.std(female_heights)
print("Point estimate for the mean population height of female patients: ", round(sample_mean,

#Lower bound for the CI of population mean (95% confidence)
lb_pop_mean_95 = sample_mean - (t_95*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (95% confidence)
ub_pop_mean_95 = sample_mean + (t_95*(sample_std/math.sqrt(n)))

#Lower bound for the CI of population mean (99% confidence)
lb_pop_mean_99 = sample_mean - (t_99*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (99% confidence)
ub_pop_mean_99 = sample_mean + (t_99*(sample_std/math.sqrt(n)))

print("The 95% Confidence Interval for the mean height of the Female population: ({}, {})".form
print("The 99% Confidence Interval for the mean height of the Female population: ({}, {})".form
```

```
Point estimate for the mean population height of female patients:  163.447
The 95% Confidence Interval for the mean height of the Female population: (163.411, 163.484)
The 99% Confidence Interval for the mean height of the Female population: (163.399, 163.495)
```

In [26]:
```
# Confidence Interval Estimation for STD of Height of Female Population

female_heights = df_female["Height_(cm)"].tolist()
n = len(female_heights)
```

```
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
N = (n-1)/n

sample_std = np.std(female_heights)
print("Point estimate for the STD of the population of female patients: ", round(math.sqrt(N)*s

#Lower bound for the CI of population STD (95% confidence)
lb_pop_std_95 = math.sqrt((n-1)/b_95)*sample_std
#Upper bound for the CI of population STD (95% confidence)
ub_pop_std_95 = math.sqrt((n-1)/a_95)*sample_std

#Lower bound for the CI of population STD (99% confidence)
lb_pop_std_99 = math.sqrt((n-1)/b_99)*sample_std
#Upper bound for the CI of population STD (99% confidence)
ub_pop_std_99 = math.sqrt((n-1)/a_99)*sample_std

print("The 95% Confidence Interval for the population STD of female patients: ({}, {})".format(
print("The 99% Confidence Interval for the population STD of female patients: ({}, {})".format(
```

```
Point estimate for the STD of the population of female patients:  7.462
The 95% Confidence Interval for the population STD of female patients: (7.436, 7.488)
The 99% Confidence Interval for the population STD of female patients: (7.428, 7.496)
```

In [27]:

```
# Population Variance of Heights of Female Patients

female_heights = df_female["Height_(cm)"].tolist()
n = len(female_heights)
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
N = (n-1)/n


sample_var = np.var(female_heights)
print("Point estimate for the population variance of height of female patients: ", round(N*samp

#Lower bound for the CI of population variance (95% confidence)
lb_pop_var_95 = ((n-1)*sample_var)/b_95
#Upper bound for the CI of population variance (95% confidence)
ub_pop_var_95 = ((n-1)*sample_var)/a_95

#Lower bound for the CI of population variance (99% confidence)
lb_pop_var_99 = ((n-1)*sample_var)/b_99
#Upper bound for the CI of population variance (99% confidence)
ub_pop_var_99 = ((n-1)*sample_var)/a_99

print("The 95% Confidence Interval for the population variance of height of female patients: ({
print("The 99% Confidence Interval for the population variance of height of female patients: ({
```

```
Point estimate for the population variance of height of female patients:  55.678
The 95% Confidence Interval for the population variance of height of female patients: (55.295, 5
6.066)
The 99% Confidence Interval for the population variance of height of female patients: (55.175, 5
6.189)
```

1.2.4 Confidence Interval for Dfference between means of Male and Female Heights

If $X_1, X_2, \ldots X_n \sim N(\mu_1, \sigma^2)$ and $Y_1, Y_2, \ldots Y_m \sim N(\mu_2, \sigma^2)$ are independent samples, then a $(1-\alpha)100\%$ confidence interval for the difference in population means $\mu_1 - \mu_2$ is:

$$(\overline{X} - \overline{Y}) \pm t_{\alpha/2, n+m-2}\ S_p \sqrt{\frac{1}{n} + \frac{1}{m}}$$

where S_p is the pooled standard deviation, given by $S_p = \sqrt{\frac{(n-1)S_X^2 + (m-1)S_Y^2}{n+m-2}}$

In [28]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
df_male = dfo.loc[dfo["Sex"]=="Male", :]
male_heights = df_male["Height_(cm)"].tolist()
female_heights = df_female["Height_(cm)"].tolist()
n_male = len(male_heights)
n_female = len(female_heights)
S_male = np.std(male_heights)
S_female = np.std(female_heights)
mean_male = np.mean(male_heights)
mean_female = np.mean(female_heights)
print("Average height of male residents:", round(mean_male,3))
print("Average height of female residents:", round(mean_female,3))
print("(Sample) STD of heights of male residents:", round(S_male,3))
print("(Sample) STD of heights of female residents:", round(S_female,3))
print("Number of male residents (n1) = {} and number of female residents (n2) = {}".format(n_ma
print("Ratio of standard deviations (male/female):", round(S_male/S_female,3))
```

```
Average height of male residents: 178.34
Average height of female residents: 163.447
(Sample) STD of heights of male residents: 7.808
(Sample) STD of heights of female residents: 7.462
Number of male residents (n1) = 148658 and number of female residents (n2) = 160196
Ratio of standard deviations (male/female): 1.046
```

As the ratio of sample standard deviation between the male and female heights is 1.05, which is less than 2, we use the pooled t-interval for CI of difference of means.

In [29]:
```
var_male = np.var(male_heights)
var_female = np.var(female_heights)
Sp = math.sqrt(((n_male-1)*var_male + (n_female-1)*var_female)/(n_male+ n_female - 2)) #pooled
t_95 = stats.t.ppf(q = 1-(0.05/2), df = n_male+n_female-2)
t_99 = stats.t.ppf(q=1-(0.01/2), df = n_male+n_female-2)

lb_95 = (mean_male - mean_female) - (t_95 * Sp * math.sqrt((1/n_male) + (1/n_female)))
ub_95 = (mean_male - mean_female) + (t_95 * Sp * math.sqrt((1/n_male) + (1/n_female)))

lb_99 = (mean_male - mean_female) - (t_99 * Sp * math.sqrt((1/n_male) + (1/n_female)))
ub_99 = (mean_male - mean_female) + (t_99 * Sp * math.sqrt((1/n_male) + (1/n_female)))

print("The pooled standard deviation is {}.\n".format(Sp))
print("Confidence intervals for difference in population mean height between male and female re
print("The 95% Confidence Interval: ({}, {})".format(round(lb_95,3), round(ub_95,3)))
print("The 99% Confidence Interval: ({}, {})".format(round(lb_99,3), round(ub_99,3)))
```

```
The pooled standard deviation is 7.63043858750532.

Confidence intervals for difference in population mean height between male and female residents:
The 95% Confidence Interval: (14.839, 14.946)
The 99% Confidence Interval: (14.822, 14.963)
```

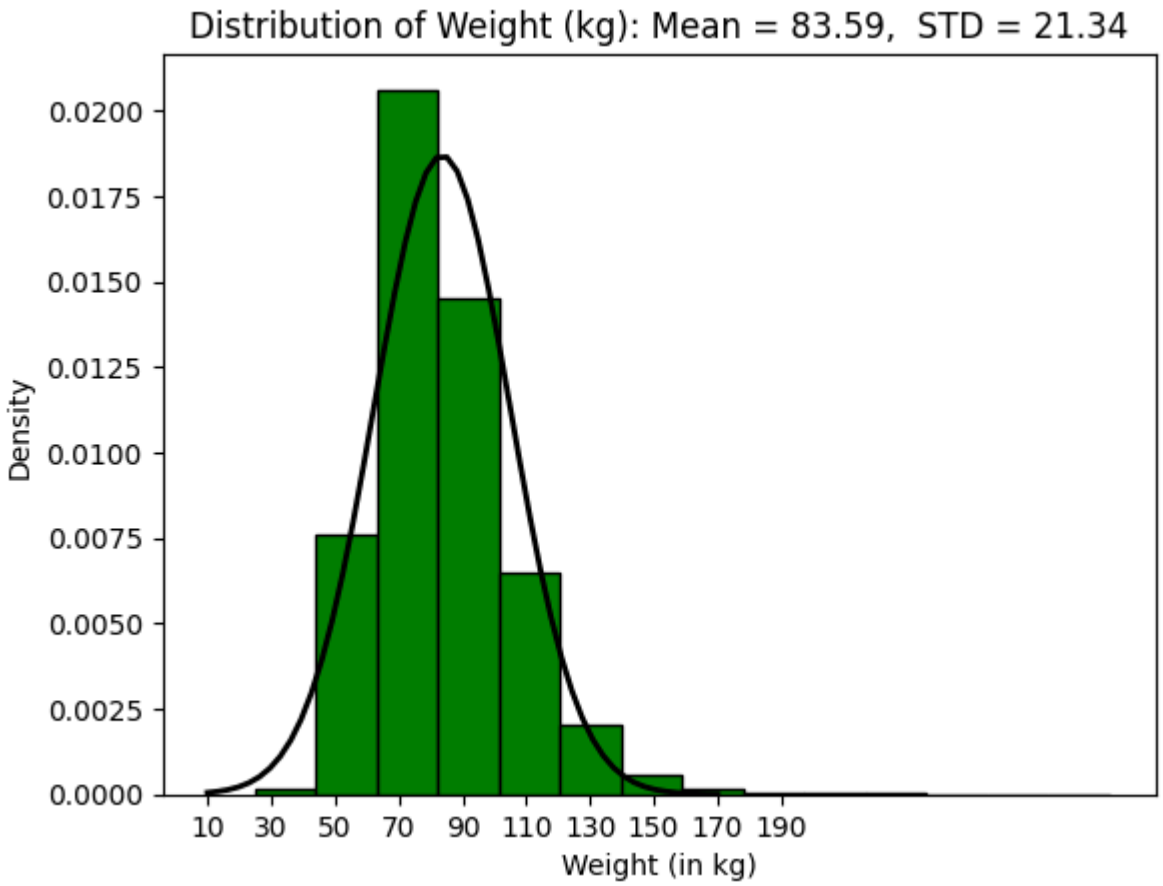
1.3 Analysis of Weight Data

1.3.1 Distribution of Weight (kg) of all Residents

Weight of the US residents is approximately normally distributed with mean 83.59 kg and standard deviation 21.34 kg.

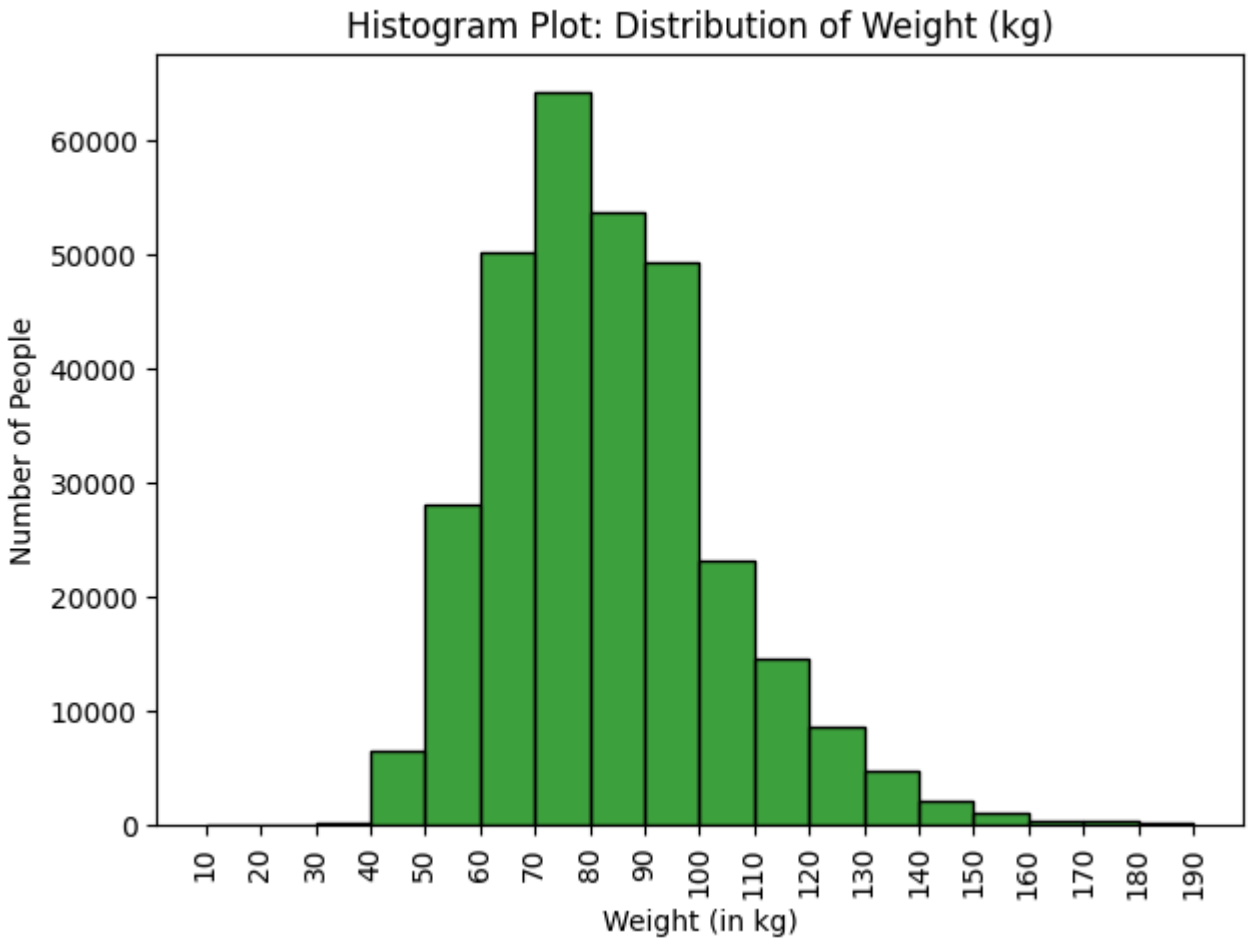
In [30]:
```
from scipy.stats import norm
import matplotlib.pyplot as plt

data = dfo['Weight_(kg)'].tolist()

# Fit a normal distribution to the data:
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=14, density=True, color='green', ec='k')

# Plot the PDF.

x = np.linspace(10, 170, 50)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
xr = [i for i in range(10, 201, 20)]

title = "Distribution of Weight (kg): Mean = %.2f,  STD = %.2f" % (mu, std)
plt.xticks(xr)
plt.title(title)
plt.ylabel("Density")
plt.xlabel("Weight (in kg)")
```

Out[30]:
```
Text(0.5, 0, 'Weight (in kg)')
```



In [31]:
```
plt.figure(figsize=(7,5))
plt.title("Histogram Plot: Distribution of Weight (kg)")
ax = sns.histplot(dfo["Weight_(kg)"], bins=[x for x in range(10, 191, 10)], color="green")
plt.ylabel("Number of People")
plt.xlabel("Weight (in kg)")
plt.xticks([x for x in range(10, 191, 10)], rotation=90)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na o
ption is deprecated and will be removed in a future version. Convert inf values to NaN before op
erating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Confidence Interval Estimation for Mean Weight of Population

In [32]:
```
n = dfo.shape[0]
# Note that the value of parameter q is 1 - (alpha/2). For 95% CI, we have alpha = 0.05
t_95 = stats.t.ppf(q = 0.975, df=n-1)
t_99 = stats.t.ppf(q = 0.995, df=n-1)
print("The t-values are as follows:")
print("t_95 = ", t_95)
print("t_99 = ", t_99)
```

```
The t-values are as follows:
t_95 =  1.959971665476839
t_99 =  2.575845222371783
```

In [33]:
```
# Mean Weight of Population

weights = dfo["Weight_(kg)"].tolist()
n = len(weights)

sample_mean = np.mean(weights)
sample_std = np.std(weights)

print("Point estimate for population mean weight (Sample mean): ", round(sample_mean, 3))

#Lower bound for the CI of population mean (95% confidence)
lb_pop_mean_95 = sample_mean - (t_95*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (95% confidence)
ub_pop_mean_95 = sample_mean + (t_95*(sample_std/math.sqrt(n)))

#Lower bound for the CI of population mean (99% confidence)
```

```
lb_pop_mean_99 = sample_mean - (t_99*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (99% confidence)
ub_pop_mean_99 = sample_mean + (t_99*(sample_std/math.sqrt(n)))

print("The 95% Confidence Interval for the mean weight (in kg) of the population: ({}, {})".for
print("The 99% Confidence Interval for the mean weight (in kg) of the population: ({}, {})".for
```

```
Point estimate for population mean weight (Sample mean):  83.589
The 95% Confidence Interval for the mean weight (in kg) of the population: (83.513, 83.664)
The 99% Confidence Interval for the mean weight (in kg) of the population: (83.49, 83.688)
```

In [34]:
```
# Variance of Weight of Population

weights = dfo["Weight_(kg)"].tolist()
n = len(weights)
N = (n-1)/n
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
sample_var = np.var(weights)
print("Point estimate for the population variance of weight (Sample variance): ", round(N*sampl

#Lower bound for the CI of population variance (95% confidence)
lb_pop_var_95 = ((n-1)*sample_var)/b_95
#Upper bound for the CI of population variance (95% confidence)
ub_pop_var_95 = ((n-1)*sample_var)/a_95

#Lower bound for the CI of population variance (99% confidence)
lb_pop_var_99 = ((n-1)*sample_var)/b_99
#Upper bound for the CI of population variance (99% confidence)
ub_pop_var_99 = ((n-1)*sample_var)/a_99

print("The 95% Confidence Interval for the population variance of weight: ({}, {})".format(roun
print("The 99% Confidence Interval for the population variance of weight: ({}, {})".format(roun
```

```
Point estimate for the population variance of weight (Sample variance):  455.53
The 95% Confidence Interval for the population variance of weight: (453.268, 457.812)
The 99% Confidence Interval for the population variance of weight: (452.559, 458.531)
```

In [35]:
```
# Population Standard Deviation of Weight

weights = dfo["Weight_(kg)"].tolist()
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
n = len(weights)
N = (n-1)/n
sample_std = np.std(weights)
print("Point estimate for the standard deviation of weight (Sample STD): ", round(math.sqrt(N)*

#Lower bound for the CI of population STD (95% confidence)
lb_pop_std_95 = math.sqrt((n-1)/b_95)*sample_std
#Upper bound for the CI of population STD (95% confidence)
ub_pop_std_95 = math.sqrt((n-1)/a_95)*sample_std

#Lower bound for the CI of population STD (99% confidence)
lb_pop_std_99 = math.sqrt((n-1)/b_99)*sample_std
#Upper bound for the CI of population STD (99% confidence)
ub_pop_std_99 = math.sqrt((n-1)/a_99)*sample_std
```

```
print("The 95% Confidence Interval for the population STD of weight: ({}, {})".format(round(lb_
print("The 99% Confidence Interval for the population STD of weight: ({}, {})".format(round(lb_
```

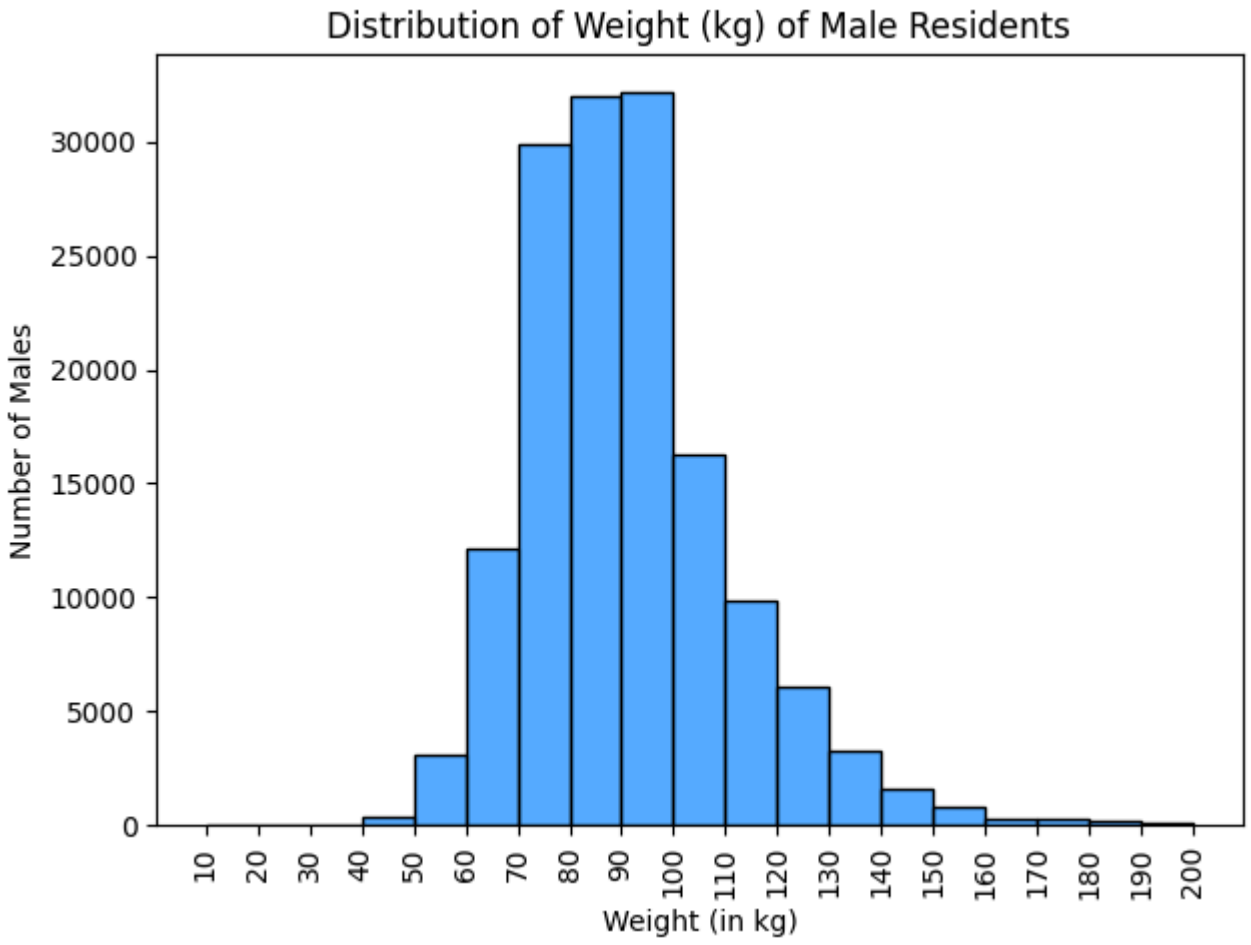
```
Point estimate for the standard deviation of weight (Sample STD):  21.343
The 95% Confidence Interval for the population STD of weight: (21.29, 21.397)
The 99% Confidence Interval for the population STD of weight: (21.273, 21.413)
```

1.3.2 Weight Distribution by Gender: Male Residents

In [36]:
```
df_male = dfo.loc[dfo["Sex"]=="Male", :]
```

In [37]:
```
#Weight of Male Residents

plt.figure(figsize=(7,5))
plt.title("Distribution of Weight (kg) of Male Residents")
ax = sns.histplot(df_male["Weight_(kg)"], bins=[x for x in range(10,201,10)], color='dodgerblue
plt.ylabel("Number of Males")
plt.xlabel("Weight (in kg)")
plt.xticks([x for x in range(10, 201, 10)], rotation=90)
plt.show()
```

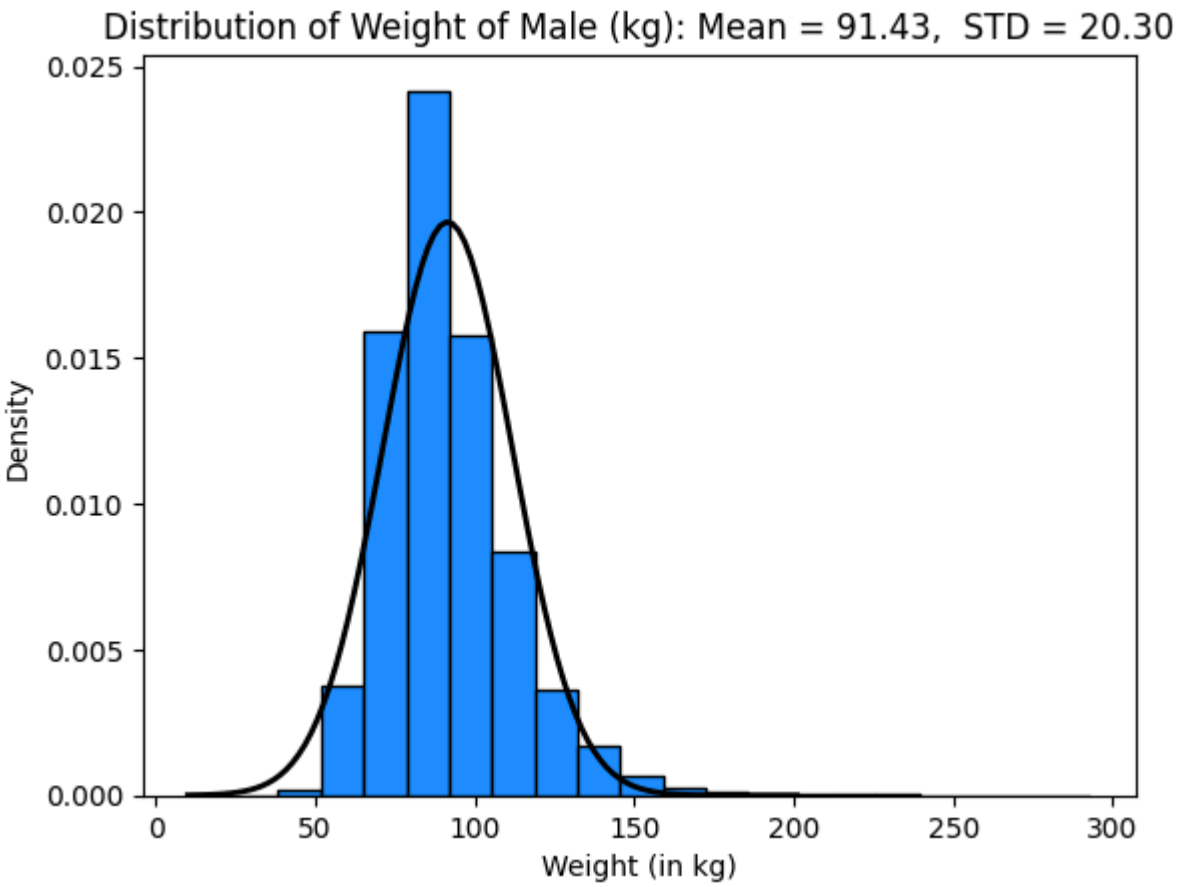
```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na o
ption is deprecated and will be removed in a future version. Convert inf values to NaN before op
erating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [38]:
```
from scipy.stats import norm

data = df_male['Weight_(kg)'].tolist()
```

```
# Fit a normal distribution to the data:
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=20, density=True, ec='k', color='dodgerblue')


# Plot the PDF.

x = np.linspace(10, 201, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
title = "Distribution of Weight of Male (kg): Mean = %.2f,  STD = %.2f" % (mu, std)
plt.title(title)
plt.xlabel("Weight (in kg)")
plt.ylabel("Density")
```

Out[38]: Text(0, 0.5, 'Density')



In [39]:
```
# Confidence Interval Estimation for Mean Weight of Male Population

male_weights = df_male["Weight_(kg)"].tolist()
n = len(male_weights)
t_95 = stats.t.ppf(q = 0.975, df=n-1)
t_99 = stats.t.ppf(q = 0.995, df=n-1)

sample_mean = np.mean(male_weights)
sample_std = np.std(male_weights)
print("Point estimate for population mean weight of males (Sample mean): ", round(sample_mean,

#Lower bound for the CI of population mean (95% confidence)
lb_pop_mean_95 = sample_mean - (t_95*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (95% confidence)
```

```
ub_pop_mean_95 = sample_mean + (t_95*(sample_std/math.sqrt(n)))

#Lower bound for the CI of population mean (99% confidence)
lb_pop_mean_99 = sample_mean - (t_99*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (99% confidence)
ub_pop_mean_99 = sample_mean + (t_99*(sample_std/math.sqrt(n)))

print("The 95% Confidence Interval for the mean weight of the Male population: ({}, {})".format
print("The 99% Confidence Interval for the mean weight of the Male population: ({}, {})".format
```

```
Point estimate for population mean weight of males (Sample mean):  91.432
The 95% Confidence Interval for the mean weight of the Male population: (91.329, 91.535)
The 99% Confidence Interval for the mean weight of the Male population: (91.297, 91.568)
```

In [40]:
```
# Confidence Interval Estimation for STD of Weight of Male Population

male_weights = df_male["Weight_(kg)"].tolist()
n = len(male_weights)
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
N = (n-1)/n

sample_std = np.std(male_weights)
print("Point estimate for the STD of weight of the population of male patients: ", round(math.s

#Lower bound for the CI of population STD (95% confidence)
lb_pop_std_95 = math.sqrt((n-1)/b_95)*sample_std
#Upper bound for the CI of population STD (95% confidence)
ub_pop_std_95 = math.sqrt((n-1)/a_95)*sample_std

#Lower bound for the CI of population STD (99% confidence)
lb_pop_std_99 = math.sqrt((n-1)/b_99)*sample_std
#Upper bound for the CI of population STD (99% confidence)
ub_pop_std_99 = math.sqrt((n-1)/a_99)*sample_std

print("The 95% Confidence Interval for the population STD of weight of male patients: ({}, {})"
print("The 99% Confidence Interval for the population STD of weight of male patients: ({}, {})"
```

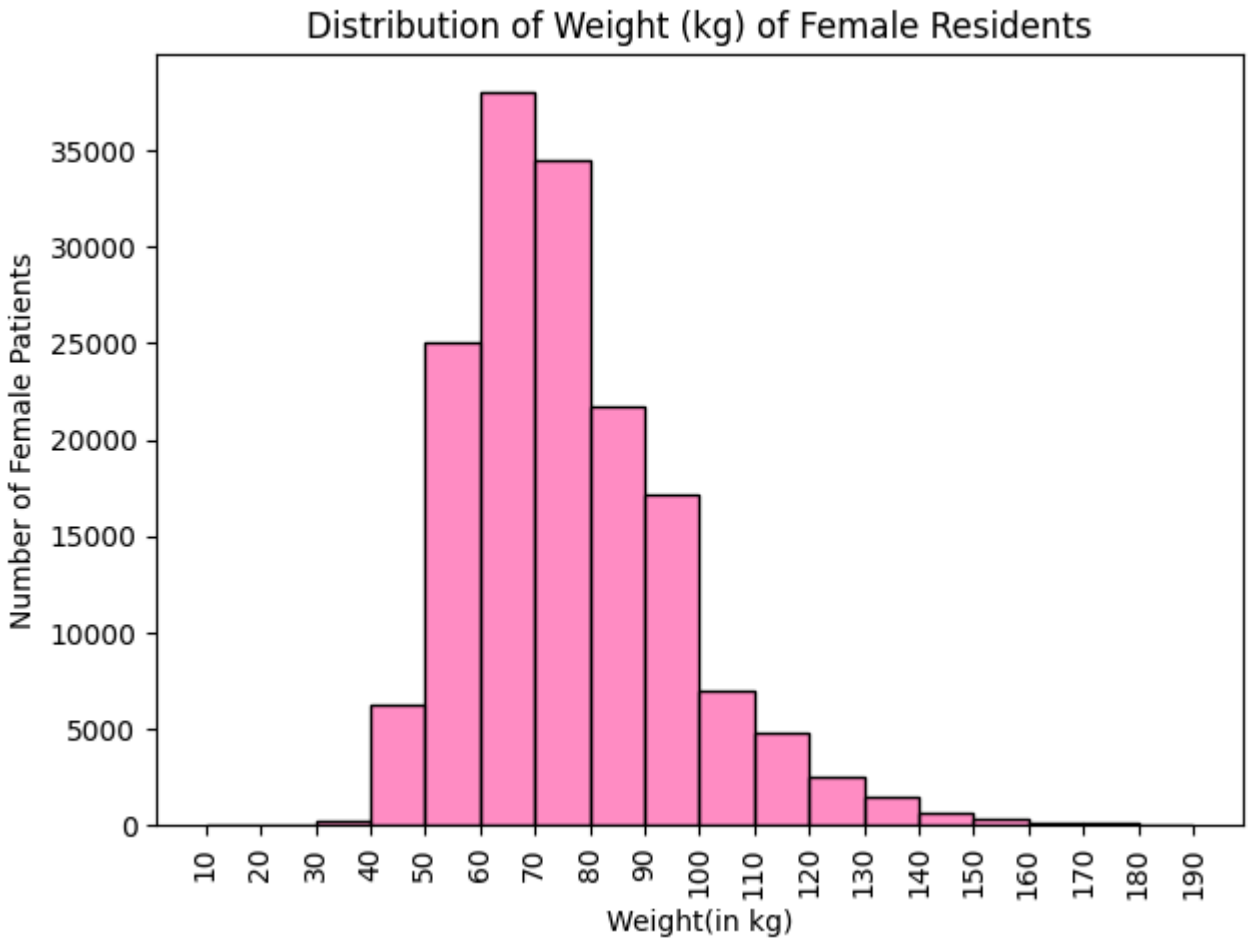
```
Point estimate for the STD of weight of the population of male patients:  20.296
The 95% Confidence Interval for the population STD of weight of male patients: (20.223, 20.369)
The 99% Confidence Interval for the population STD of weight of male patients: (20.201, 20.392)
```

1.3.3 Weight Distribution by Gender: Female Patients

In [41]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
```

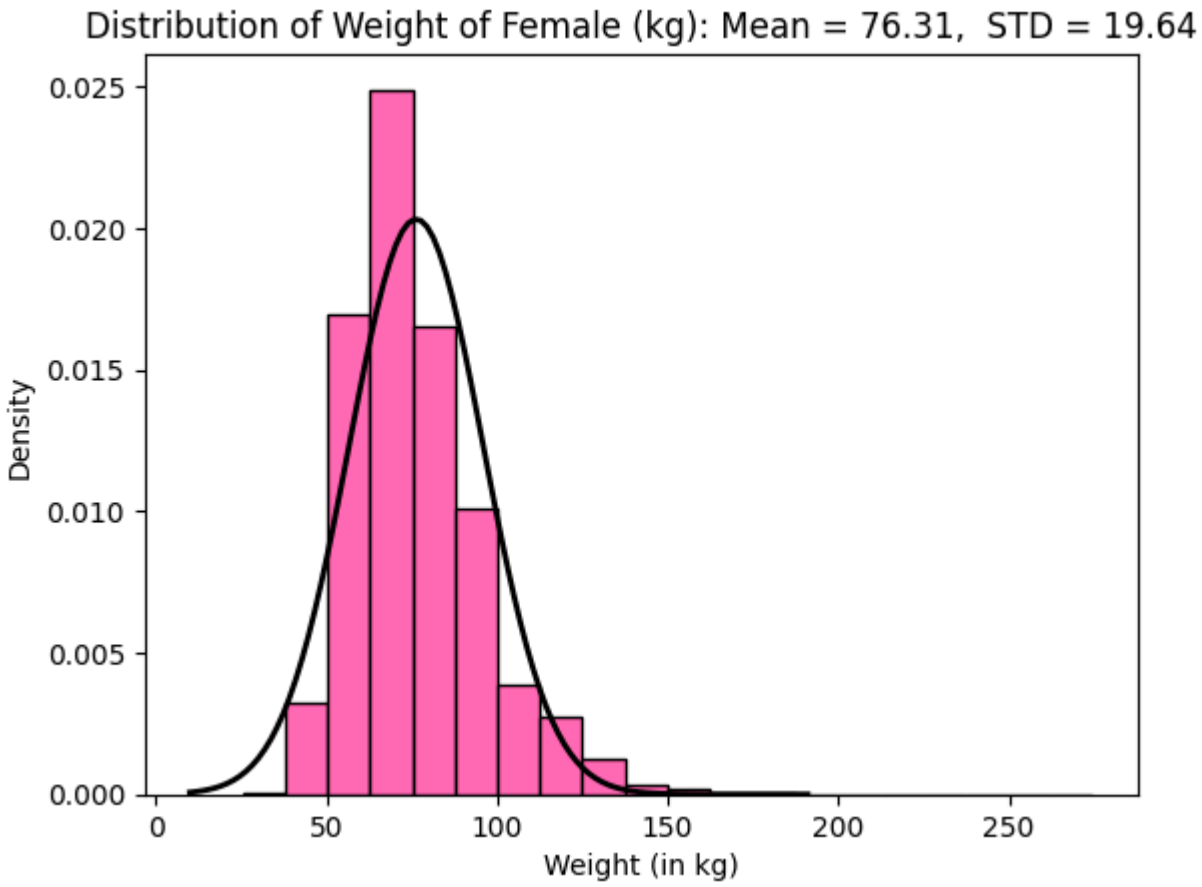
In [42]:
```
#Weight of Female Residents

plt.figure(figsize=(7,5))
plt.title("Distribution of Weight (kg) of Female Residents")
ax = sns.histplot(df_female["Weight_(kg)"], bins=[x for x in range(10,191,10)], color='hotpink'
plt.ylabel("Number of Female Patients")
plt.xlabel("Weight(in kg)")
plt.xticks([x for x in range(10, 191, 10)], rotation=90)
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na o
ption is deprecated and will be removed in a future version. Convert inf values to NaN before op
erating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [43]:
```python
from scipy.stats import norm

data = df_female['Weight_(kg)'].tolist()

# Fit a normal distribution to the data:
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=20, density=True, ec='k', color='hotpink')


# Plot the PDF.

x = np.linspace(10, 191, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
title = "Distribution of Weight of Female (kg): Mean = %.2f,  STD = %.2f" % (mu, std)
plt.title(title)
plt.xlabel("Weight (in kg)")
plt.ylabel("Density")
```

Out[43]:
```
Text(0, 0.5, 'Density')
```



In [44]:
```
# Confidence Interval Estimation for Mean Weight of Female Population

female_weights = df_female["Weight_(kg)"].tolist()
n = len(female_weights)
t_95 = stats.t.ppf(q = 0.975, df=n-1)
t_99 = stats.t.ppf(q = 0.995, df=n-1)

sample_mean = np.mean(female_weights)
sample_std = np.std(female_weights)
print("Point estimate for population mean weight of females (Sample mean): ", round(sample_mean

#Lower bound for the CI of population mean (95% confidence)
lb_pop_mean_95 = sample_mean - (t_95*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (95% confidence)
ub_pop_mean_95 = sample_mean + (t_95*(sample_std/math.sqrt(n)))

#Lower bound for the CI of population mean (99% confidence)
lb_pop_mean_99 = sample_mean - (t_99*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (99% confidence)
ub_pop_mean_99 = sample_mean + (t_99*(sample_std/math.sqrt(n)))

print("The 95% Confidence Interval for the mean weight of the Female population: ({}, {})".form
print("The 99% Confidence Interval for the mean weight of the Female population: ({}, {})".form
```

```
Point estimate for population mean weight of females (Sample mean):  76.31
The 95% Confidence Interval for the mean weight of the Female population: (76.214, 76.406)
The 99% Confidence Interval for the mean weight of the Female population: (76.184, 76.436)
```

In [45]:
```
# Confidence Interval Estimation for STD of Weight of Female Population

female_weights = df_female["Weight_(kg)"].tolist()
n = len(female_weights)
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
```

```
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
N = (n-1)/n

sample_std = np.std(female_weights)
print("Point estimate for the STD of weight of the population of female patients: ", round(math

#Lower bound for the CI of population STD (95% confidence)
lb_pop_std_95 = math.sqrt((n-1)/b_95)*sample_std
#Upper bound for the CI of population STD (95% confidence)
ub_pop_std_95 = math.sqrt((n-1)/a_95)*sample_std

#Lower bound for the CI of population STD (99% confidence)
lb_pop_std_99 = math.sqrt((n-1)/b_99)*sample_std
#Upper bound for the CI of population STD (99% confidence)
ub_pop_std_99 = math.sqrt((n-1)/a_99)*sample_std

print("The 95% Confidence Interval for the population STD of weight of female patients: ({}, {}
print("The 99% Confidence Interval for the population STD of weight of female patients: ({}, {}
```

```
Point estimate for the STD of weight of the population of female patients:  19.645
The 95% Confidence Interval for the population STD of weight of female patients: (19.577, 19.71
3)
The 99% Confidence Interval for the population STD of weight of female patients: (19.556, 19.73
5)
```

In [46]:

```
# Population Variance of Weight of Female Residents

female_weights = df_female["Weight_(kg)"].tolist()
n = len(female_weights)
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)
N = (n-1)/n


sample_var = np.var(female_weights)
print("Point estimate for the population variance of weight of female patients: ", round(N*samp

#Lower bound for the CI of population variance (95% confidence)
lb_pop_var_95 = ((n-1)*sample_var)/b_95
#Upper bound for the CI of population variance (95% confidence)
ub_pop_var_95 = ((n-1)*sample_var)/a_95

#Lower bound for the CI of population variance (99% confidence)
lb_pop_var_99 = ((n-1)*sample_var)/b_99
#Upper bound for the CI of population variance (99% confidence)
ub_pop_var_99 = ((n-1)*sample_var)/a_99

print("The 95% Confidence Interval for the population variance of weight of female patients: ({
print("The 99% Confidence Interval for the population variance of weight of female patients: ({
```

```
Point estimate for the population variance of weight of female patients:  385.918
The 95% Confidence Interval for the population variance of weight of female patients: (383.262,
388.607)
The 99% Confidence Interval for the population variance of weight of female patients: (382.431,
389.456)
```

1.4 Confidence Interval for Ratio of Variances of BMI (Female/Male)

If $X_1, X_2, \ldots X_n \sim N(\mu_X, \sigma_X^2)$ and $Y_1, Y_2, \ldots Y_m \sim N(\mu_Y, \sigma_Y^2)$ are independent samples, then a $(1-\alpha)100\%$ confidence interval for the ratio of population variances $\frac{\sigma_X^2}{\sigma_Y^2}$ is:

$$\left(\frac{1}{F_{\alpha/2,\ n-1,\ m-1}} \frac{S_X^2}{S_Y^2},\ F_{\alpha/2,\ m-1,\ n-1} \frac{S_X^2}{S_Y^2} \right)$$

In [47]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
df_male = dfo.loc[dfo["Sex"]=="Male", :]
male_bmi = df_male["BMI"].tolist()
female_bmi = df_female["BMI"].tolist()
n_male = len(male_bmi)
n_female = len(female_bmi)
var_male = np.var(male_bmi)
var_female = np.var(female_bmi)
print("(Sample) Variance of BMI of male residents (S1^2):", var_male)
print("(Sample) Variance of BMI of female residents (S2^2):", var_female)
```

```
(Sample) Variance of BMI of male residents (S1^2): 34.61067391198052
(Sample) Variance of BMI of female residents (S2^2): 49.88939959635516
```

In [48]:
```
F1_95 = stats.f.ppf(q=1-0.025, dfn=n_female-1, dfd=n_male-1)
F2_95 = stats.f.ppf(q=1-0.025, dfn=n_male-1, dfd=n_female-1)

F1_99 = stats.f.ppf(q=1-(0.01/2), dfn=n_female-1, dfd=n_male-1)
F2_99 = stats.f.ppf(q=1-(0.01/2), dfn=n_male-1, dfd=n_female-1)

print("The F-values are:")
print(F1_99, F2_99)
```

```
The F-values are:
1.0132065892557114 1.0132037633143864
```

In [49]:
```
lb_95 = (1/F1_95)*(var_female/var_male)
ub_95 = F2_95*(var_female/var_male)

lb_99 = (1/F1_99)*(var_female/var_male)
ub_99 = F2_99*(var_female/var_male)

print("Confidence intervals for ratio of variances of BMI of female and male residents:")
print("The 95% Confidence Interval: ({}, {})".format(round(lb_95,3), round(ub_95,3)))
print("The 99% Confidence Interval: ({}, {})".format(round(lb_99,3), round(ub_99,3)))
```

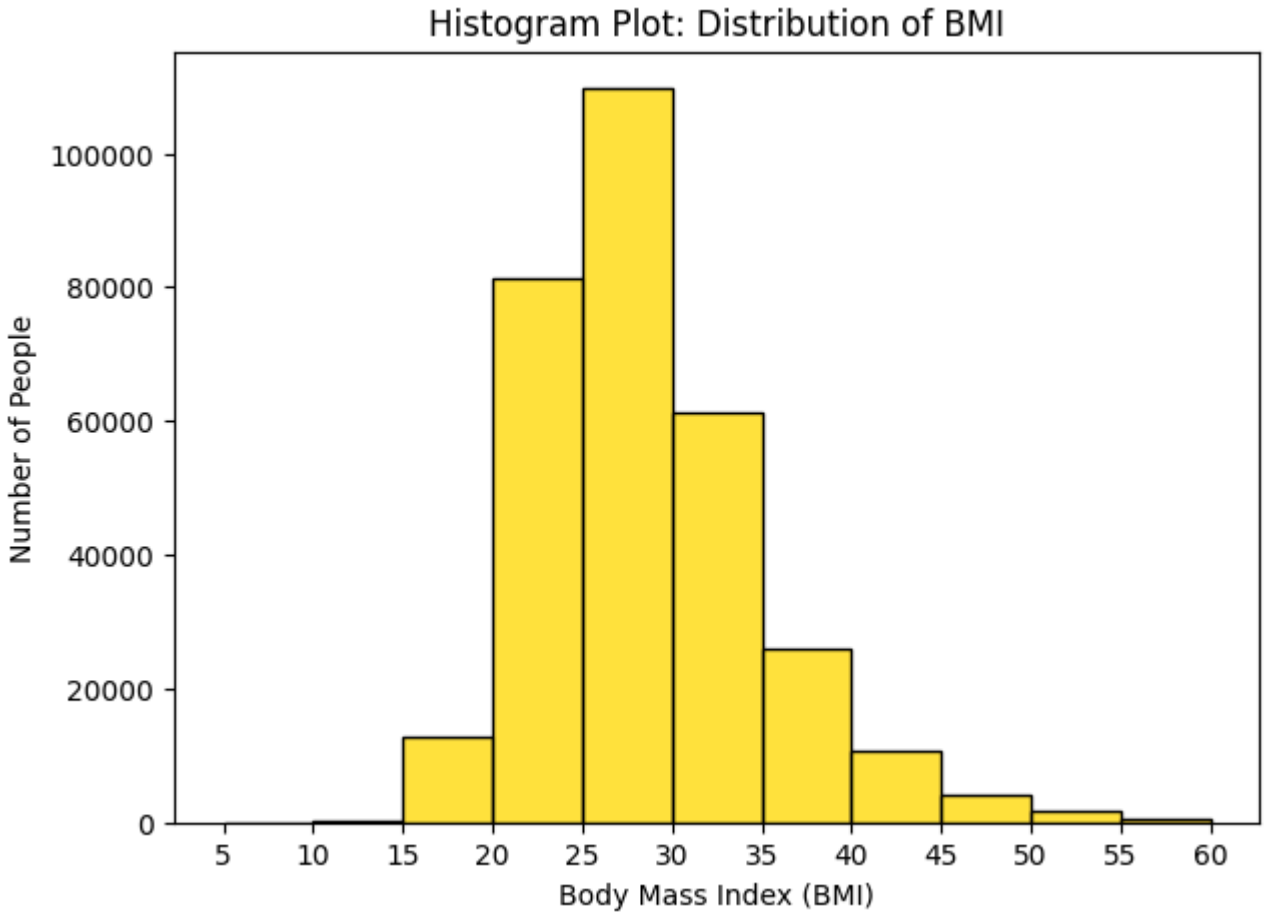
```
Confidence intervals for ratio of variances of BMI of female and male residents:
The 95% Confidence Interval: (1.427, 1.456)
The 99% Confidence Interval: (1.423, 1.46)
```

1.5 Distribution of BMI

In [50]:
```
plt.figure(figsize=(7,5))
plt.title("Histogram Plot: Distribution of BMI")
ax = sns.histplot(dfo["BMI"], bins=[x for x in range(5, 61, 5)], color="gold")
plt.ylabel("Number of People")
```

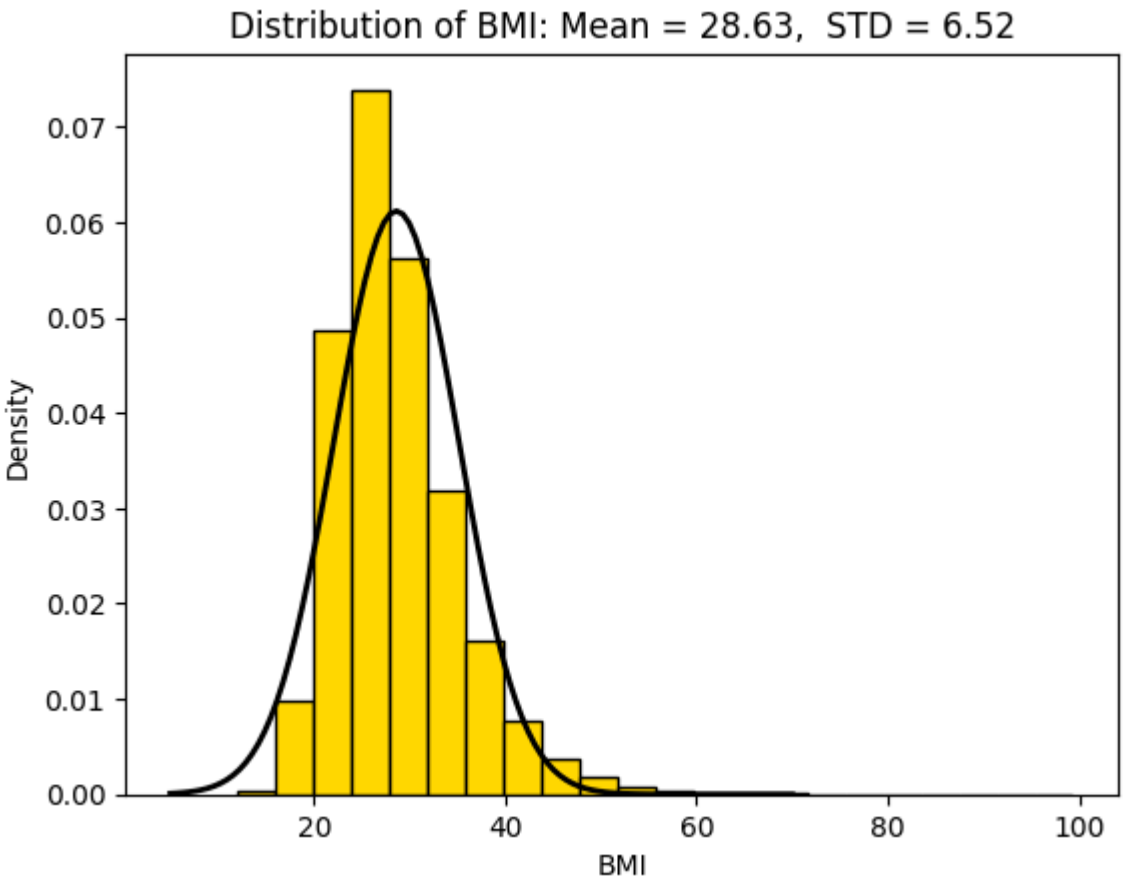
```
plt.xlabel("Body Mass Index (BMI)")
plt.xticks([x for x in range(5, 61, 5)])
plt.show()
```

```
/opt/conda/lib/python3.10/site-packages/seaborn/_oldcore.py:1119: FutureWarning: use_inf_as_na o
ption is deprecated and will be removed in a future version. Convert inf values to NaN before op
erating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [51]:
```
from scipy.stats import norm

data = dfo['BMI'].tolist()

# Fit a normal distribution to the data:
mu, std = norm.fit(data)

# Plot the histogram.
plt.hist(data, bins=22, density=True, ec='k', color='gold')


# Plot the PDF.

x = np.linspace(5, 70, 100)
p = norm.pdf(x, mu, std)
plt.plot(x, p, 'k', linewidth=2)
title = "Distribution of BMI: Mean = %.2f,  STD = %.2f" % (mu, std)
plt.title(title)
plt.xlabel("BMI")
plt.ylabel("Density")
```

Out[51]:
```
Text(0, 0.5, 'Density')
```



In [52]:
```
# Confidence Interval Estimation for Mean of Population BMI

bmi_data = dfo["BMI"].tolist()
n = len(bmi_data)

t_95 = stats.t.ppf(q = 0.975, df=n-1)
t_99 = stats.t.ppf(q = 0.995, df=n-1)

sample_mean = np.mean(bmi_data)
sample_std = np.std(bmi_data)

print("Point estimate for population mean BMI (Sample mean): ", round(sample_mean, 3))

#Lower bound for the CI of population mean (95% confidence)
lb_pop_mean_95 = sample_mean - (t_95*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (95% confidence)
ub_pop_mean_95 = sample_mean + (t_95*(sample_std/math.sqrt(n)))

#Lower bound for the CI of population mean (99% confidence)
lb_pop_mean_99 = sample_mean - (t_99*(sample_std/math.sqrt(n)))
#Upper bound for the CI of population mean (99% confidence)
ub_pop_mean_99 = sample_mean + (t_99*(sample_std/math.sqrt(n)))

print("The 95% Confidence Interval for the mean BMI of the population: ({}, {})".format(round(l
print("The 99% Confidence Interval for the mean BMI of the population: ({}, {})".format(round(l
```

```
Point estimate for population mean BMI (Sample mean):  28.626
The 95% Confidence Interval for the mean BMI of the population: (28.603, 28.649)
The 99% Confidence Interval for the mean BMI of the population: (28.596, 28.656)
```

In [53]:
```
# Confidence Interval Estimation for Variance of Population BMI

bmi_data = dfo["BMI"].tolist()
```

```
n = len(bmi_data)
a_95 = stats.chi2.ppf(q=(0.05/2), df=n-1)
b_95 = stats.chi2.ppf(q= 1-(0.05/2), df=n-1)
a_99 = stats.chi2.ppf(q= 0.01/2, df=n-1)
b_99 = stats.chi2.ppf(q= 1-(0.01/2), df=n-1)

sample_var = np.var(bmi_data)
print("Point estimate for the variance of population BMI: ", round(sample_var, 3))

#Lower bound for the CI of population variance (95% confidence)
lb_pop_var_95 = ((n-1)/b_95)*sample_var
#Upper bound for the CI of population variance (95% confidence)
ub_pop_var_95 = ((n-1)/a_95)*sample_var

#Lower bound for the CI of population variance (99% confidence)
lb_pop_var_99 = ((n-1)/b_99)*sample_var
#Upper bound for the CI of population variance (99% confidence)
ub_pop_var_99 = ((n-1)/a_99)*sample_var

print("The 95% Confidence Interval for the variance of population BMI: ({}, {})".format(round(l
print("The 99% Confidence Interval for the variance of population BMI: ({}, {})".format(round(l
```

```
Point estimate for the variance of population BMI:  42.541
The 95% Confidence Interval for the variance of population BMI: (42.329, 42.754)
The 99% Confidence Interval for the variance of population BMI: (42.263, 42.821)
```

1.6 Determining the Sample Size

Q1. How many female patients should we survey in order to be 95% confident that their estimated average height will be within 3 cm of the mean height, given that the previous data indicates a normal distribution of heights ranging from 135 cm to 195 cm?

Crude method: In this method, we simply replace the t-value that depends on n with a z-value that (does not because as n increases, the t-distribution approaches the standard normal distribution). Thus,

$$n \approx \frac{(z_{\alpha/2})^2 S^2}{E^2}$$

Here, E is the margin of error. Also, as the \textbf{empirical rule} states that approximately 95% of the measurements lie in the interval $\mu \pm 2\sigma$, we estimate σ by $S = \frac{Range}{4}$, where $Range = (\mu + 2\sigma) - (\mu - 2\sigma) = 4\sigma$ is the length of the interval.

In [54]:
```
#1. Crude Method
z_95 = stats.norm.ppf(q = 0.975)
E = 3
S = (195-135)/4       # Sample STD estimate
n = (((z_95)**2)*(S**2))/(E**2)
n
```

Out[54]:
```
96.03647051735314
```

∴ Through the crude method, we have that a sample size of approximately **96 or larger** is recommended to obtain an estimated average female height that we are 95% confident is within 3 cm of the true mean female height.

Iterative Method: A more accurate method to estimate the sample size is to iteratively evaluate the formula since the t value also depends on n. We start with an initial guess for n and plugin the formula

$$n = \frac{(t_{\alpha/2,\ n-1})^2 S^2}{E^2}$$

and iteratively solve for n.

In [55]:
```
# 2. Iterative method

n = 30    # Initial guess
E = 3     # Margin of Error
S = (195-135)/4        # Sample STD estimate = Range/4
for i in range(8):
    t = stats.t.ppf(q=1-(0.05/2), df=n-1)
    new_n = ((t**2)*(S**2))/(E**2)
    print("Iteration {}: \t Assumed n = {} \t Calculated n = {}".format((i+1), round(n,3), roun
    n = new_n
```

```
Iteration 1:     Assumed n = 30          Calculated n = 104.574
Iteration 2:     Assumed n = 104.574     Calculated n = 98.32
Iteration 3:     Assumed n = 98.32       Calculated n = 98.47
Iteration 4:     Assumed n = 98.47       Calculated n = 98.466
Iteration 5:     Assumed n = 98.466      Calculated n = 98.466
Iteration 6:     Assumed n = 98.466      Calculated n = 98.466
Iteration 7:     Assumed n = 98.466      Calculated n = 98.466
Iteration 8:     Assumed n = 98.466      Calculated n = 98.466
```

∴ A sample size of approximately **98 or larger** is recommended to obtain an estimated average female height that we are 95% confident is within 3 cm of the true mean female height.

Q2. How many patients should we survey in order to be 95% confident that their estimated average BMI will be within $5kg/m^2$ of the mean BMI, given that the previous data indicates a normal distribution of BMIs ranging from $10kg/m^2$ to $60kg/m^2$?

In [56]:
```
#1. Crude Method
z_95 = stats.norm.ppf(q = 0.975)
E = 5
S = (60-10)/4       # Sample STD estimate
n = (((z_95)**2)*(S**2))/(E**2)
n
```

Out[56]:
```
24.009117629338284
```

∴ Through the Crude method, a sample size of approximately **24 or larger** is recommended to obtain an estimated average BMI that we are 95% confident is within $5kg/m^2$ of the true mean BMI.

In [57]:
```
# 2. Iterative method

n = 30    # Initial guess
E = 5    # Margin of Error
S = (60-10)/4        # Sample STD estimate = Range/4
for i in range(8):
    t = stats.t.ppf(q=1-(0.05/2), df=n-1)
    new_n = ((t**2)*(S**2))/(E**2)
```

```
    print("Iteration {}: \t Assumed n = {} \t Calculated n = {}".format((i+1), round(n,3), roun
    n = new_n
```

```
Iteration 1:     Assumed n = 30          Calculated n = 26.144
Iteration 2:     Assumed n = 26.144      Calculated n = 26.495
Iteration 3:     Assumed n = 26.495      Calculated n = 26.458
Iteration 4:     Assumed n = 26.458      Calculated n = 26.462
Iteration 5:     Assumed n = 26.462      Calculated n = 26.462
Iteration 6:     Assumed n = 26.462      Calculated n = 26.462
Iteration 7:     Assumed n = 26.462      Calculated n = 26.462
Iteration 8:     Assumed n = 26.462      Calculated n = 26.462
```

∴ A sample size of approximately **27 or larger** is recommended to obtain an estimated average BMI that we are 95% confident is within $5kg/m^2$ of the true mean BMI.

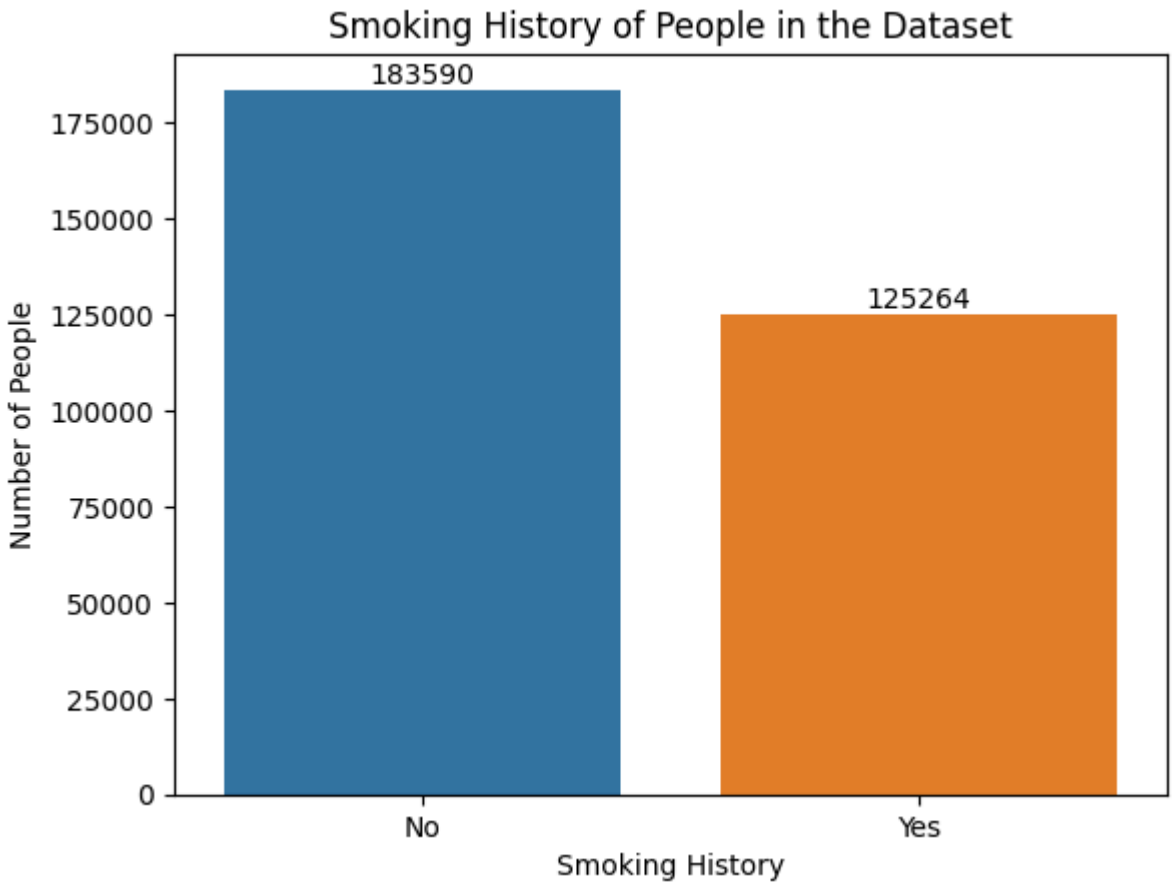
1.7 Confidence Interval for Population Proportion

As we have a large number of samples, a $100(1-\alpha)\%$ confidence interval for population proportion $\hat{p}$ is:

$$\hat{p} \pm z_{\frac{\alpha}{2}} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Q1. What percent of the population has a smoking history?

In [58]:
```
smoke = dfo["Smoking_History"]

ax = sns.countplot(data=dfo, x="Smoking_History", order=smoke.value_counts().index)
ax.bar_label(ax.containers[0])
plt.xlabel("Smoking History")
plt.ylabel("Number of People")
plt.title("Smoking History of People in the Dataset")

plt.show()
```



In [59]:
```
smoke = dfo["Smoking_History"]
non_smokers = smoke.value_counts().iloc[0]
smokers = smoke.value_counts().iloc[1]
print("In our dataset, out of {} people, {} have a smoking history.".format(smokers+non_smokers
```

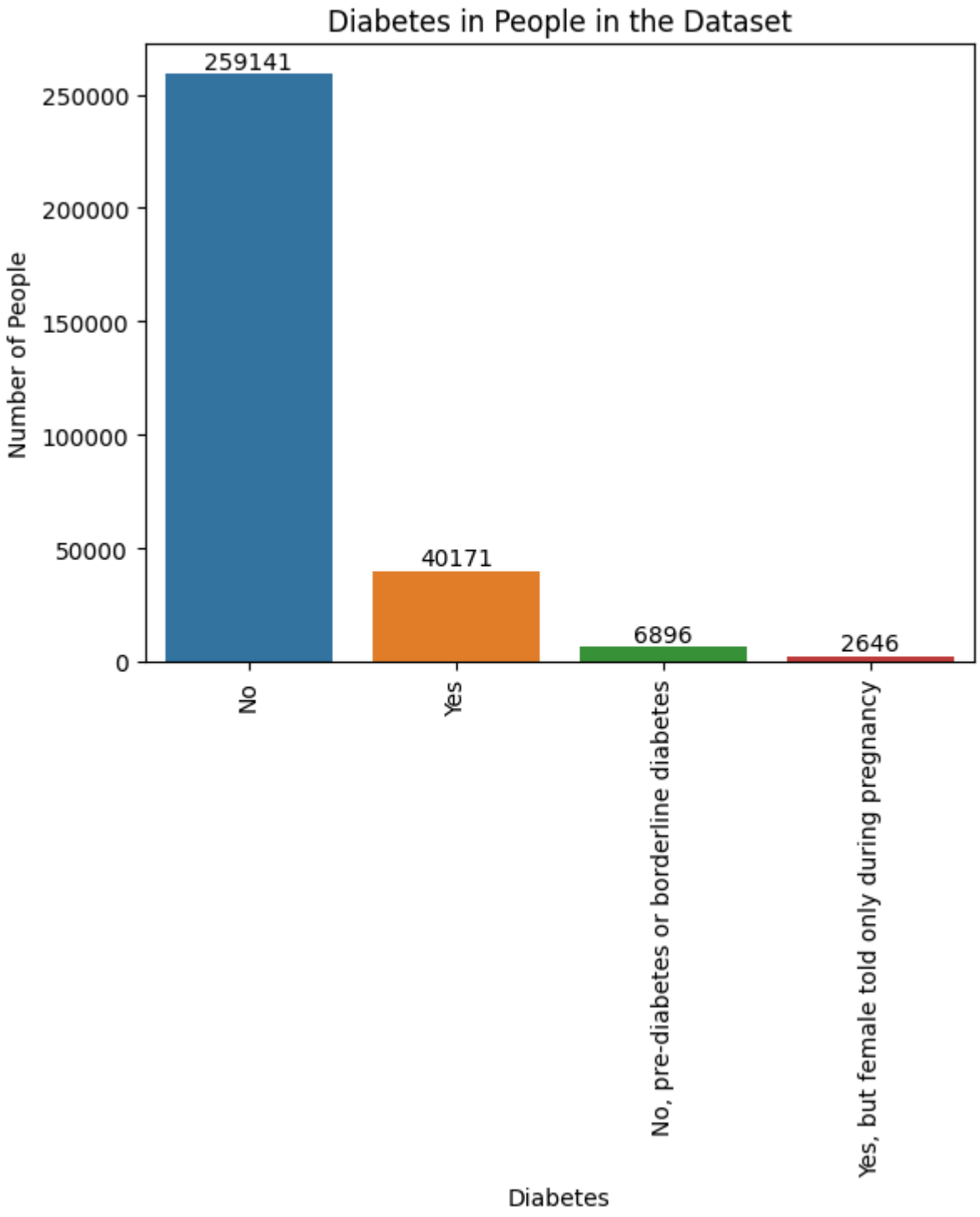
```
In our dataset, out of 308854 people, 125264 have a smoking history.
```

In [60]:
```
z_95 = stats.norm.ppf(q=0.975)
n = smokers + non_smokers
p_hat = smokers/n
lb = p_hat - (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
ub = p_hat + (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
print("We can be 95% confident that between {}% to {}% of the population have a smoking history
```

```
We can be 95% confident that between 40.385% to 40.731% of the population have a smoking histor
y.
```

Q2. What percent of the population is diabetic?

In [61]:
```
diabetes = dfo["Diabetes"]

ax = sns.countplot(data=dfo, x="Diabetes", order=diabetes.value_counts().index)
ax.bar_label(ax.containers[0])
plt.xlabel("Diabetes")
plt.ylabel("Number of People")
plt.title("Diabetes in People in the Dataset")
plt.xticks(rotation=90)

plt.show()
```



Let us treat pre-diabetic patients as diabetic, and females who reported diabetes during pregnancy only to be non-diabetic.

In [62]:
```
diabetes = dfo["Diabetes"]
non_diabetic = diabetes.value_counts().iloc[0] + diabetes.value_counts().iloc[3]
diabetic = diabetes.value_counts().iloc[1] + diabetes.value_counts().iloc[2]
total = non_diabetic + diabetic
print("In our dataset, out of {} people, {} are diabetic".format(total, diabetic))
```

```
In our dataset, out of 308854 people, 47067 are diabetic
```

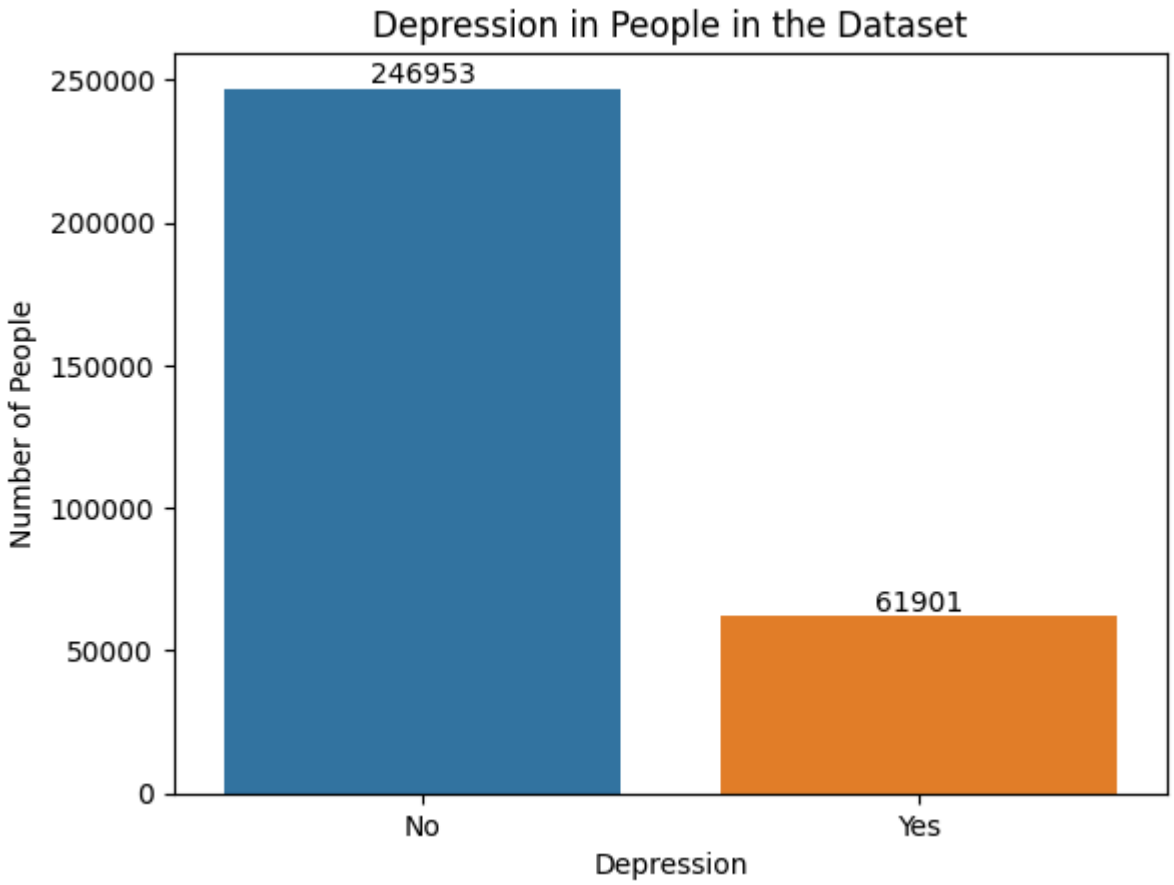
In [63]:
```
z_95 = stats.norm.ppf(q=0.975)
n = non_diabetic + diabetic
p_hat = diabetic/n
lb = p_hat - (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
```

```
ub = p_hat + (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
print("We can be 95% confident that between {}% to {}% of the population is diabetic.".format(r
```

```
We can be 95% confident that between 15.112% to 15.366% of the population is diabetic.
```

Q3. What percent of the population has depression?

In [64]:
```
depression = dfo["Depression"]

ax = sns.countplot(data=dfo, x="Depression", order=depression.value_counts().index)
ax.bar_label(ax.containers[0])
plt.xlabel("Depression")
plt.ylabel("Number of People")
plt.title("Depression in People in the Dataset")

plt.show()
```



In [65]:
```
depression_data = dfo["Depression"]
no_depression = depression_data.value_counts().iloc[0]
depression = depression_data.value_counts().iloc[1]
total = no_depression + depression
print("In our dataset, out of {} people, {} are suffering from depression".format(total, depres
```

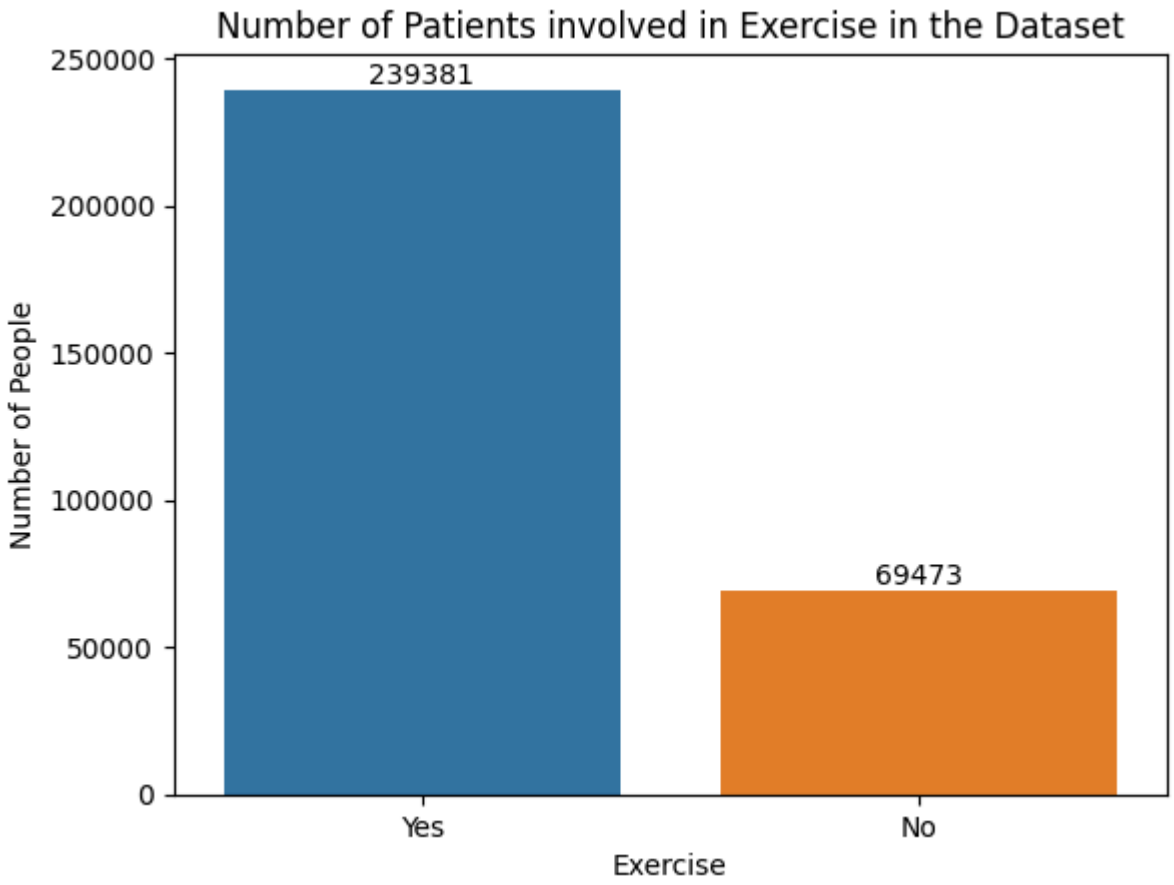
```
In our dataset, out of 308854 people, 61901 are suffering from depression
```

In [66]:
```
z_95 = stats.norm.ppf(q=0.975)
n = no_depression + depression
p_hat = depression/n
lb = p_hat - (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
ub = p_hat + (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
print("We can be 95% confident that between {}% to {}% of the population is suffering from depr
```

```
We can be 95% confident that between 19.901% to 20.183% of the population is suffering from depr
ession.
```

Q4. What percent of the population is involved in exercise or any kind of physical activities?

In [67]:
```
exercise = dfo["Exercise"]

ax = sns.countplot(data=dfo, x="Exercise", order=exercise.value_counts().index)
ax.bar_label(ax.containers[0])
plt.xlabel("Exercise")
plt.ylabel("Number of People")
plt.title("Number of Patients involved in Exercise in the Dataset")

plt.show()
```



In [68]:
```
exercise_data = dfo["Exercise"]
no_exercise = exercise_data.value_counts().iloc[0]
exercise = exercise_data.value_counts().iloc[1]
total = no_exercise + exercise
print("In our dataset, out of {} patients, {} are involved in exercise.".format(total, exercise
```

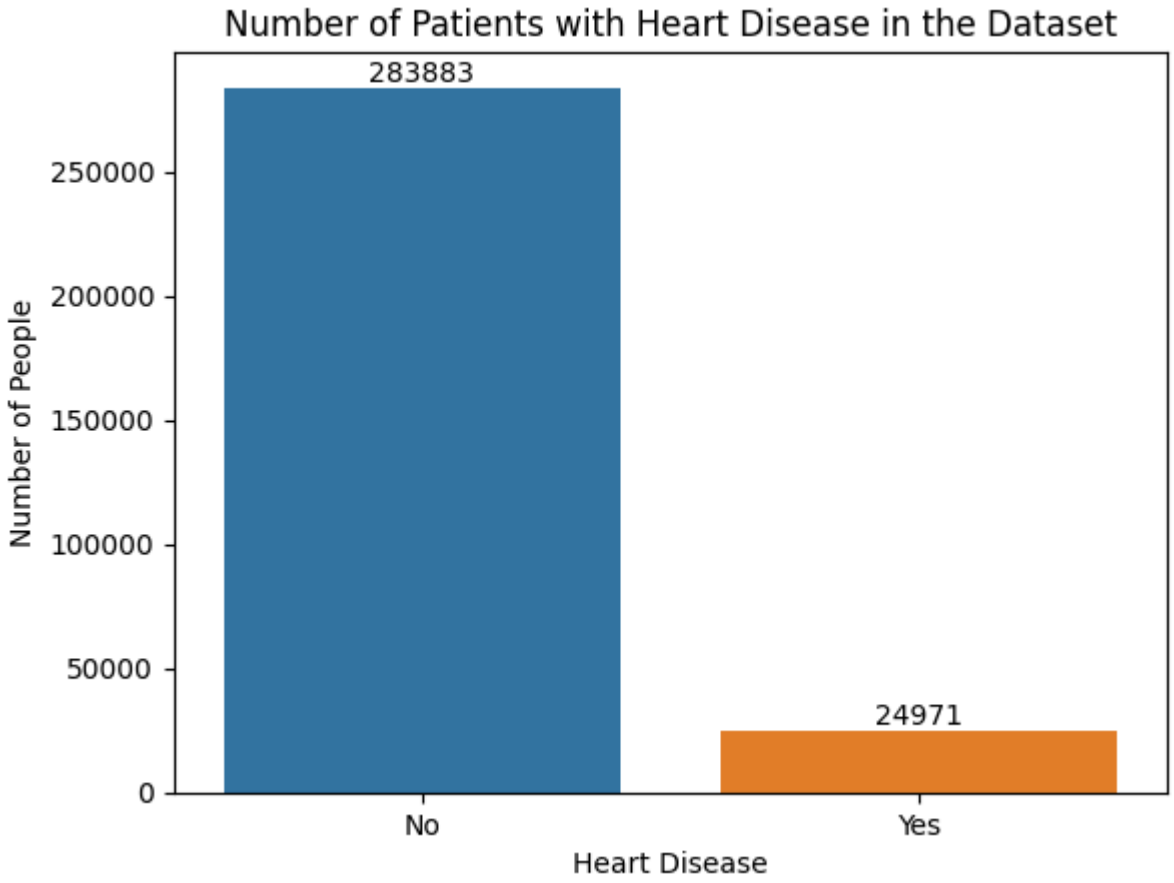
```
In our dataset, out of 308854 patients, 69473 are involved in exercise.
```

In [69]:
```
z_95 = stats.norm.ppf(q=0.975)
n = no_exercise + exercise
p_hat = exercise/n
lb = p_hat - (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
ub = p_hat + (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
print("We can be 95% confident that between {}% to {}% of the population is involved in exercis
```

We can be 95% confident that between 22.347% to 22.641% of the population is involved in exercis
e or physical activities.

Q5. What percent of the population has heart disease?

In [70]:
```
hd = dfo["Heart_Disease"]

ax = sns.countplot(data=dfo, x="Heart_Disease", order=hd.value_counts().index)
ax.bar_label(ax.containers[0])
plt.xlabel("Heart Disease")
plt.ylabel("Number of People")
plt.title("Number of Patients with Heart Disease in the Dataset")

plt.show()
```



In [71]:
```
hd_data = dfo["Heart_Disease"]
no_hd = hd_data.value_counts().iloc[0]
hd = hd_data.value_counts().iloc[1]
total = no_hd + hd
print("In our dataset, out of {} patients, {} have heart disease.".format(total, hd))
```

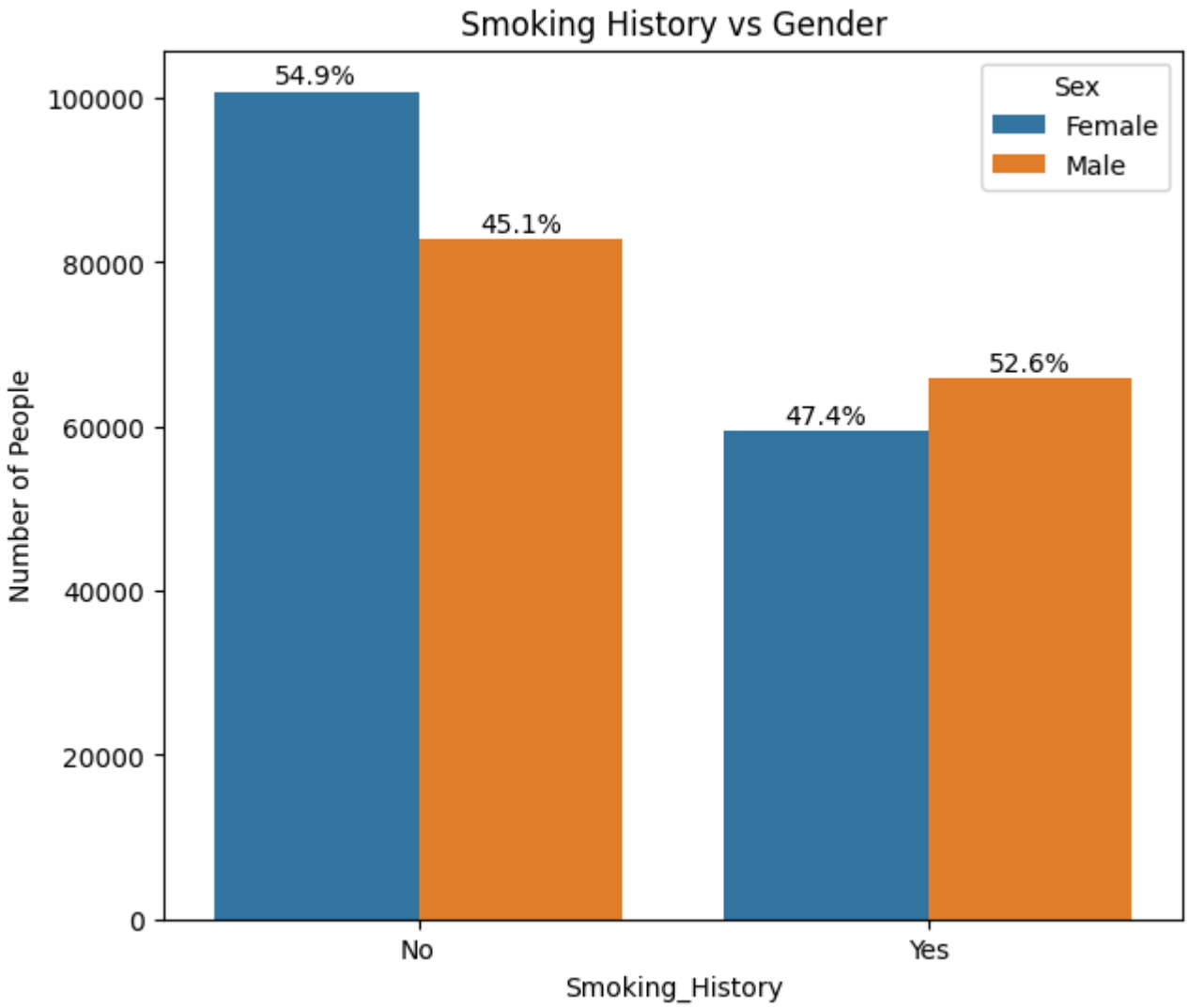
In our dataset, out of 308854 patients, 24971 have heart disease.

In [72]:
```
z_95 = stats.norm.ppf(q=0.975)
n = no_hd + hd
p_hat = hd/n
lb = p_hat - (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
ub = p_hat + (z_95 * math.sqrt((p_hat * (1-p_hat))/n))
print("We can be 95% confident that between {}% to {}% of the population has heart disease.".fo
```

We can be 95% confident that between 7.989% to 8.181% of the population has heart disease.

Q6. Difference in Proportion of Male Smokers and Female Smokers

In [73]:
```
column = 'Smoking_History'
target = 'Sex'

fig,ax = plt.subplots(figsize = (7,6))

#color = 'Set2'

#palette_color = sns.color_palette(color)

ax = sns.countplot(x = column, data=dfo, hue=target, order = dfo[column].value_counts().index)
ax.set_ylabel('Count')

offset = dfo[column].value_counts().max() * 0.005

list_bars = dfo.groupby([column,target])[column].agg(['count']).unstack().fillna(0).values

patches = ax.patches
bars_pos = 0

for i in range(dfo[target].nunique()):
    for j in range(dfo[column].nunique()):
        list_bars_col = list_bars[j]
        total_sum = list_bars_col.sum()
        value = list_bars_col[i]

        percentage = value / total_sum

        if percentage == 0:
            bars_pos += 1
            continue
        else:
            x = patches[bars_pos].get_x() + patches[j].get_width()/2
            y = patches[bars_pos].get_height() + offset
            ax.annotate('{:.1f}%'.format(percentage*100), (x, y), ha='center')
            bars_pos += 1
plt.ylabel("Number of People")
plt.title("Smoking History vs Gender")
plt.show()
```



In [74]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
df_male = dfo.loc[dfo["Sex"]=="Male", :]

male_smo = df_male["Smoking_History"]
female_smo = df_female["Smoking_History"]
male_non_smokers = male_smo.value_counts().iloc[0]
female_non_smokers = female_smo.value_counts().iloc[0]
male_smokers = male_smo.value_counts().iloc[1]
female_smokers = female_smo.value_counts().iloc[1]
n_male = len(male_smo)
n_female = len(female_smo)
p1 = male_smokers/n_male      # (sample) proportion of male smokers
p2 = female_smokers/n_female

print("Number of male smokers: {}, and number of male residents who do not smoke: {}".format(ma
print("Number of female smokers: {}, and number of female residents who do not smoke: {}".forma
print("(Sample) Proportion of male smokers (p1):", p1)
print("(Sample) Proportion of female smokers (p2):", p2)
```

```
Number of male smokers: 65854, and number of male residents who do not smoke: 82804
Number of female smokers: 59410, and number of female residents who do not smoke: 100786
(Sample) Proportion of male smokers (p1): 0.4429899500867764
(Sample) Proportion of female smokers (p2): 0.37085819870658443
```

We can say that about 44.3% of the male residents are smokers, and so are about 37.1% of the female residents.

In [75]:
```
z_95 = stats.norm.ppf(q=1-0.025)
z_99 = stats.norm.ppf(q=1-(0.01/2))
```

In [76]:
```
temp = ((p1*(1-p1))/n_male) + ((p2*(1-p2))/n_female)
lb_95 = (p1-p2) - (z_95 * math.sqrt(temp))
ub_95 = (p1-p2) + (z_95 * math.sqrt(temp))

lb_99 = (p1-p2) - (z_99 * math.sqrt(temp))
ub_99 = (p1-p2) + (z_99 * math.sqrt(temp))

print("Confidence intervals for difference in proportion of male and female smokers:")
print("The 95% Confidence Interval: ({}, {})".format(round(lb_95,3), round(ub_95,3)))
print("The 99% Confidence Interval: ({}, {})".format(round(lb_99,3), round(ub_99,3)))
```

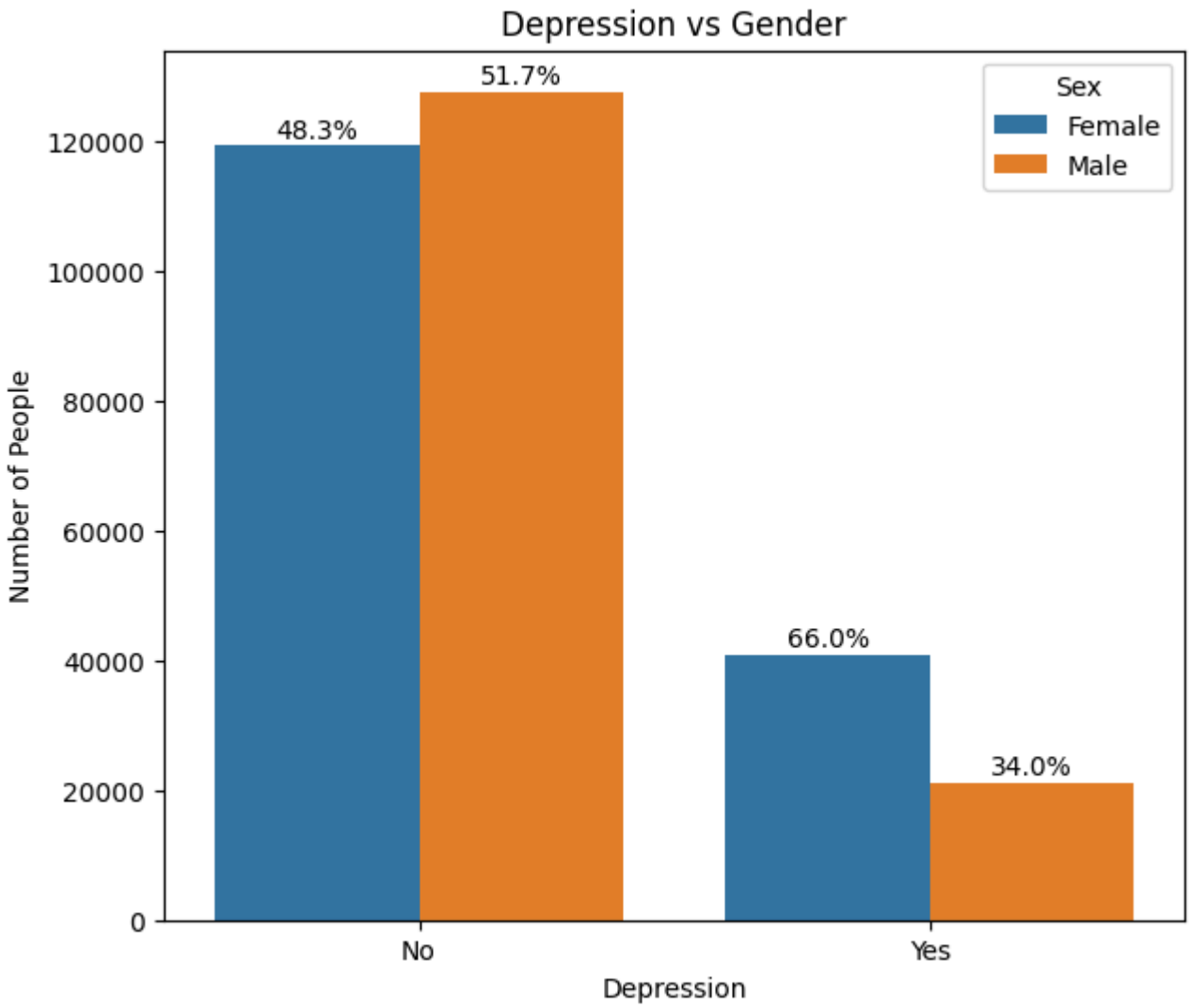
```
Confidence intervals for difference in proportion of male and female smokers:
The 95% Confidence Interval: (0.069, 0.076)
The 99% Confidence Interval: (0.068, 0.077)
```

We can be 95% confident that 6.9% to 7.6% more male residents of the US are smokers as compared to the female residents.

Q7. Difference in Proportion: Depression in Women and Men

In [77]:
```
column = 'Depression'
target = 'Sex'

fig,ax = plt.subplots(figsize = (7,6))

#color = 'Set2'

#palette_color = sns.color_palette(color)

ax = sns.countplot(x = column, data=dfo, hue=target, order = dfo[column].value_counts().index)
ax.set_ylabel('Count')

offset = dfo[column].value_counts().max() * 0.005

list_bars = dfo.groupby([column,target])[column].agg(['count']).unstack().fillna(0).values

patches = ax.patches
bars_pos = 0

for i in range(dfo[target].nunique()):
    for j in range(dfo[column].nunique()):
        list_bars_col = list_bars[j]
        total_sum = list_bars_col.sum()
        value = list_bars_col[i]

        percentage = value / total_sum

        if percentage == 0:
            bars_pos += 1
            continue
        else:
            x = patches[bars_pos].get_x() + patches[j].get_width()/2
            y = patches[bars_pos].get_height() + offset
            ax.annotate('{:.1f}%'.format(percentage*100), (x, y), ha='center')
            bars_pos += 1
```

```
plt.ylabel("Number of People")
plt.title("Depression vs Gender")
plt.show()
```



In [78]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
df_male = dfo.loc[dfo["Sex"]=="Male", :]

male_dep = df_male["Depression"]
female_dep = df_female["Depression"]
male_non_depressed = male_dep.value_counts().iloc[0]
female_non_depressed = female_dep.value_counts().iloc[0]
male_depressed = male_dep.value_counts().iloc[1]
female_depressed = female_dep.value_counts().iloc[1]
n_male = len(male_dep)
n_female = len(female_dep)
p1 = female_depressed/n_female
p2 = male_depressed/n_male      # (sample) proportion of depressed men


print("Number of depressed men: {}, and number of men who are not depressed: {}".format(male_de
print("Number of depressed women: {}, and number of women who are not depressed: {}".format(fem

print("(Sample) Proportion of depressed women (p1):", p1)
print("(Sample) Proportion of depressed men (p2):", p2)
```

```
Number of depressed men: 21056, and number of men who are not depressed: 127602
Number of depressed women: 40845, and number of women who are not depressed: 119351
(Sample) Proportion of depressed women (p1): 0.2549689130814752
(Sample) Proportion of depressed men (p2): 0.14164054406759138
```

In [79]:
```
z_95 = stats.norm.ppf(q=1-0.025)
z_99 = stats.norm.ppf(q=1-(0.01/2))
```

In [80]:
```
temp = ((p1*(1-p1))/n_male) + ((p2*(1-p2))/n_female)
lb_95 = (p1-p2) - (z_95 * math.sqrt(temp))
ub_95 = (p1-p2) + (z_95 * math.sqrt(temp))

lb_99 = (p1-p2) - (z_99 * math.sqrt(temp))
ub_99 = (p1-p2) + (z_99 * math.sqrt(temp))

print("Confidence intervals for difference in proportion of depression in female and male resid
print("The 95% Confidence Interval: ({}, {})".format(round(lb_95,3), round(ub_95,3)))
print("The 99% Confidence Interval: ({}, {})".format(round(lb_99,3), round(ub_99,3)))
```

```
Confidence intervals for difference in proportion of depression in female and male residents of
the US:
The 95% Confidence Interval: (0.111, 0.116)
The 99% Confidence Interval: (0.11, 0.117)
```

We can be 95% confident that 11.1% to 11.6% more women in the US suffer from depression as compared to men.

---

2. Hypothesis Testing on Numerical Features

1. Whether the average male height in the US was still 175.3 cm in 2021 (Reported by Wikipedia, 2015-18)

H_0: Average height of the male population in the US is 175.3 cm $(\mu = 175.3)$

H_a: Average height of the male population in the US is not 175.3 cm $(\mu \neq 175.3)$

In [81]:
```
df_male = dfo.loc[dfo["Sex"]=="Male", :]
n = df_male.shape[0]
male_heights = df_male["Height_(cm)"].tolist()
S = np.std(male_heights)
xbar = np.mean(male_heights)
mu0 = 175.3

# Let t_test denote the test statistic (t*)
t_test = (xbar - mu0)/(S/math.sqrt(n))
print("Value of the test statistic is: t* =", t_test)
```

```
Value of the test statistic is: t* = 150.0996346088145
```

In [82]:
```
print("Value of t from table at alpha = 0.05 and n-1 degrees of freedom:", stats.t.ppf(q=1-(0.0
```

```
Value of t from table at alpha = 0.05 and n-1 degrees of freedom: 1.9599799426867746
```

Rejection Region Approach:

The rejection region is $|t^*| \leq t_{\alpha/2,n-1}$, which means that we will reject the null hypothesis if $t^* > 1.96$ or $t^* < -1.96$.

Clearly, we have $t^* = 150.1 > 1.96$, so we will reject the null hypothesis H_0.

p-Value Approach:

p-value for two-tailed test: $p = 2P(t \geq |t^*|)$

Note: stats.t.sf(t, *df) returns the area under the curve of the t-distribution to the right of $t^*. That is, it returns the value of* P(t \geq t^*).$

In [83]:
```
print("The p-value is: ", stats.t.sf(t_test, df=n-1))
```

```
The p-value is:  0.0
```

Clearly, the p-value $\approx 0 < \alpha = 0.05$. So, we reject the null hypothesis.

2. Whether the average female height in the US is greater than 161.3 cm (which was the value reported in Wikipedia in 2015-18)

H_0: Average height of the female population in the US has not increased from 161.3 cm $(\mu \leq 161.3)$

H_a: Average height of the female population in the US has increased from 161.3 cm $(\mu > 161.3)$

In [84]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
female_heights = df_female["Height_(cm)"].tolist()
n = len(female_heights)
S = np.std(female_heights)
xbar = np.mean(female_heights)
mu0 = 161.3

# Let t_test denote the test statistic (t*)
t_test = (xbar - mu0)/(S/math.sqrt(n))
print("Value of the test statistic is: t* =", t_test)
```

```
Value of the test statistic is: t* = 115.17133660829386
```

In [85]:
```
print("Value of t from table at alpha = 0.05 and n-1 degrees of freedom:", stats.t.ppf(q=1-0.05
```

```
Value of t from table at alpha = 0.05 and n-1 degrees of freedom: 1.6448631389712798
```

Rejection Region Approach:

The rejection region is $|t^*| \geq t_{\alpha/2,n-1}$, which means that we will reject the null hypothesis if $t^* \geq 1.96$.

Clearly, we have $t^* = 115.2 > 1.96$, so we will reject the null hypothesis H_0.

p-Value Approach:

p-value for right-tailed test: $p = P(t \geq t^*)$

In [86]:
```
print("The p-value is: ", stats.t.sf(t_test, df=n-1))
```

```
The p-value is:  0.0
```

Clearly, the p-value $\approx 0 < \alpha = 0.05$. So, we reject the null hypothesis.

3. Is the variance in BMI is lesser than $40 kg/m^2$?

H_0: $\sigma^2 \geq 40$

H_a: $\sigma^2 < 40$

In [87]:
```
bmi_data = dfo["BMI"].tolist()
n = len(bmi_data)
sigma0 = 40
var = np.var(bmi_data)
ts = ((n-1)*var)/(sigma0)
print("Value of the test statistic:", ts)
```

```
Value of the test statistic: 328469.49572981586
```

In [88]:
```
var
```

Out[88]:
```
42.540560814344154
```

Critical value: χ^2_L at $1 - \alpha$ and $n - 1$ degrees of freedom.

In [89]:
```
print("Critical value:", stats.chi2.ppf(q=0.05, df=n-1))
```

```
Critical value: 307561.3772508502
```

Rejection Region Approach

We can reject H_0 if $\chi^2_{test} < \chi^2_L$, where χ^2_{test} is the test statistic. However, here we have that $\chi^2_{test} > \chi^2_L$. Hence, we fail to reject the null hypothesis.

p-Value Approach

p-Value for left-tailed test is $p = P(\chi^2 \leq \chi^2_{test})$.

In [90]:
```
stats.chi2.cdf(ts, df=n-1)
```

Out[90]:
```
1.0
```

Clearly, $p \approx 1 > 0.05 = \alpha$, so we fail to reject the null hypothesis.

We do not have enough evidence to claim that the US population's BMI variability is less than 40 kg/m^2. A greater variance in BMI indicates that the individuals have a wide range of body sizes and shapes. This could include both underweight and severely obese individuals.

4. Are men in the US more than 14.8 cm (5.8 inches) taller than women on average?

This might be, for example, relevant to the fashion industry in a sense that the apparels manufactured for men and women would have a difference of 14.8 cm in length on average.

Let μ_{men} and μ_{women} denote the mean population height of men and women respectively.

H_0 : Men in the US are not more than 14.8 cm taller than women on average $(\mu_{men} - \mu_{women} \leq 14.8)$

H_a : Men in the US are more than 14.8 cm taller than women on average $(\mu_{men} - \mu_{women} > 14.8)$

This is a right-tailed test.

In [91]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
df_male = dfo.loc[dfo["Sex"]=="Male", :]
male_heights = df_male["Height_(cm)"].tolist()
female_heights = df_female["Height_(cm)"].tolist()
n_male = len(male_heights)
n_female = len(female_heights)
S_male = np.std(male_heights)        # Sample STD of height
S_female = np.std(female_heights)
mean_male = np.mean(male_heights)    # Sample mean height
mean_female = np.mean(female_heights)
print("Average height of male residents:", round(mean_male,3))
print("Average height of female residents:", round(mean_female,3))
print("(Sample) STD of heights of male residents:", round(S_male,3))
print("(Sample) STD of heights of female residents:", round(S_female,3))
print("Number of male residents (n1) = {} and number of female residents (n2) = {}".format(n_ma
print("Ratio of standard deviations (male/female):", round(S_male/S_female,3))
```

```
Average height of male residents: 178.34
Average height of female residents: 163.447
(Sample) STD of heights of male residents: 7.808
(Sample) STD of heights of female residents: 7.462
Number of male residents (n1) = 148658 and number of female residents (n2) = 160196
Ratio of standard deviations (male/female): 1.046
```

We may assume that the population variances of heights of men and women are nearly equal (since the ratio of standard deviations is less than 2). Hence, we use the pooled standard deviation.

In [92]:
```
var_male = np.var(male_heights)
var_female = np.var(female_heights)
Sp = math.sqrt(((n_male-1)*var_male + (n_female-1)*var_female)/(n_male+ n_female - 2)) #pooled
print("Pooled STD is:", Sp)

t_test = ((mean_male - mean_female) - 14.8)/(Sp * math.sqrt((1/n_male)+(1/n_female)))
print("The test statistic is:", t_test)
```

```
Pooled STD is: 7.63043858750532
The test statistic is: 3.367629774323511
```

Rejection Region Approach

We reject the null hypothesis if $t_{test} \geq t_{\alpha,\ n_1+n_2-2}$

In [93]:
```
t_crit = stats.t.ppf(q = 1-0.05, df = n_male+n_female-2)
print("Critical t-value at alpha = 0.05:", t_crit)
```

```
Critical t-value at alpha = 0.05: 1.6448585606211454
```

Clearly, $t_{test} \geq t_{\alpha,\ n_1+n_2-2}$, so we reject H_0.

p-Value Approach

p-value for right-tailed test is given by $p = P(t \geq t_{test})$.

In [94]:
```
p = 1-stats.t.cdf(t_test, df=n_male+n_female-2)
print("p-value:", p)
```

```
p-value: 0.00037913292433877643
```

Clearly, $p = 0.0004 < 0.05 = \alpha$, so we reject the null hypothesis.

5. Ratio of Variances: Weights of Men and Women

We want to test the claim that men have more variability in weights than women. Let σ_1^2 and σ_2^2 denote the population variance of weight of men and women respectively.

H_0: Men do not have more variability in weights than women $(\sigma_1^2 \leq \sigma_2^2)$

H_a: Men have more variability in weights than women $(\sigma_1^2 > \sigma_2^2)$

In [95]:
```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
df_male = dfo.loc[dfo["Sex"]=="Male", :]
male_weights = df_male["Weight_(kg)"].tolist()
female_weights = df_female["Weight_(kg)"].tolist()
n_male = len(male_weights)
n_female = len(female_weights)
S_male = np.std(male_weights)        # Sample STD of weight of men
S_female = np.std(female_weights)
var_male = np.var(male_weights)
var_female = np.var(female_weights)
mean_male = np.mean(male_weights)    # Sample mean height
mean_female = np.mean(female_weights)
print("Average weight of male residents:", round(mean_male,3))
print("Average weight of female residents:", round(mean_female,3))
print("(Sample) STD of weights of male residents:", round(S_male,3))
print("(Sample) STD of weights of female residents:", round(S_female,3))
print("Number of male residents (n1) = {} and number of female residents (n2) = {}".format(n_ma
print("Ratio of variances (male/female):", round(var_male/var_female,3))
```

```
Average weight of male residents: 91.432
Average weight of female residents: 76.31
(Sample) STD of weights of male residents: 20.296
(Sample) STD of weights of female residents: 19.645
Number of male residents (n1) = 148658 and number of female residents (n2) = 160196
Ratio of variances (male/female): 1.067
```

In [96]:
```
F_test = var_male/var_female
F_test
```

Out[96]:
```
1.067396132845912
```

Rejection Region Approach

For level α with $df_1 = n_{male} - 1$ and $df_2 = n_{female} - 1$, we reject H_0 if $F_{test} \geq F_{\alpha,\ df_1,\ df_2}$

In [97]:
```
df1 = n_male-1
df2 = n_female-1
F_crit = stats.f.ppf(1-0.05, dfn=df1, dfd=df2)
F_crit
```

Out[97]:
```
1.0084116742595286
```

Clearly, $F_{test} \geq F_{\alpha,\ df_1,\ df_2}$, so we reject H_0.

p-Value Approach

p = $P(F > F_{test})$

In [98]:
```
p = 1-stats.f.cdf(x=F_test, dfn = df1, dfd = df2)
print("p-value:", p)
```

```
p-value: 1.1102230246251565e-16
```

6. Diabetes is More Common in Men than in Women

Let p_1 denote the proportion of diabetic men and p_2 denote the proportion of diabetic women.

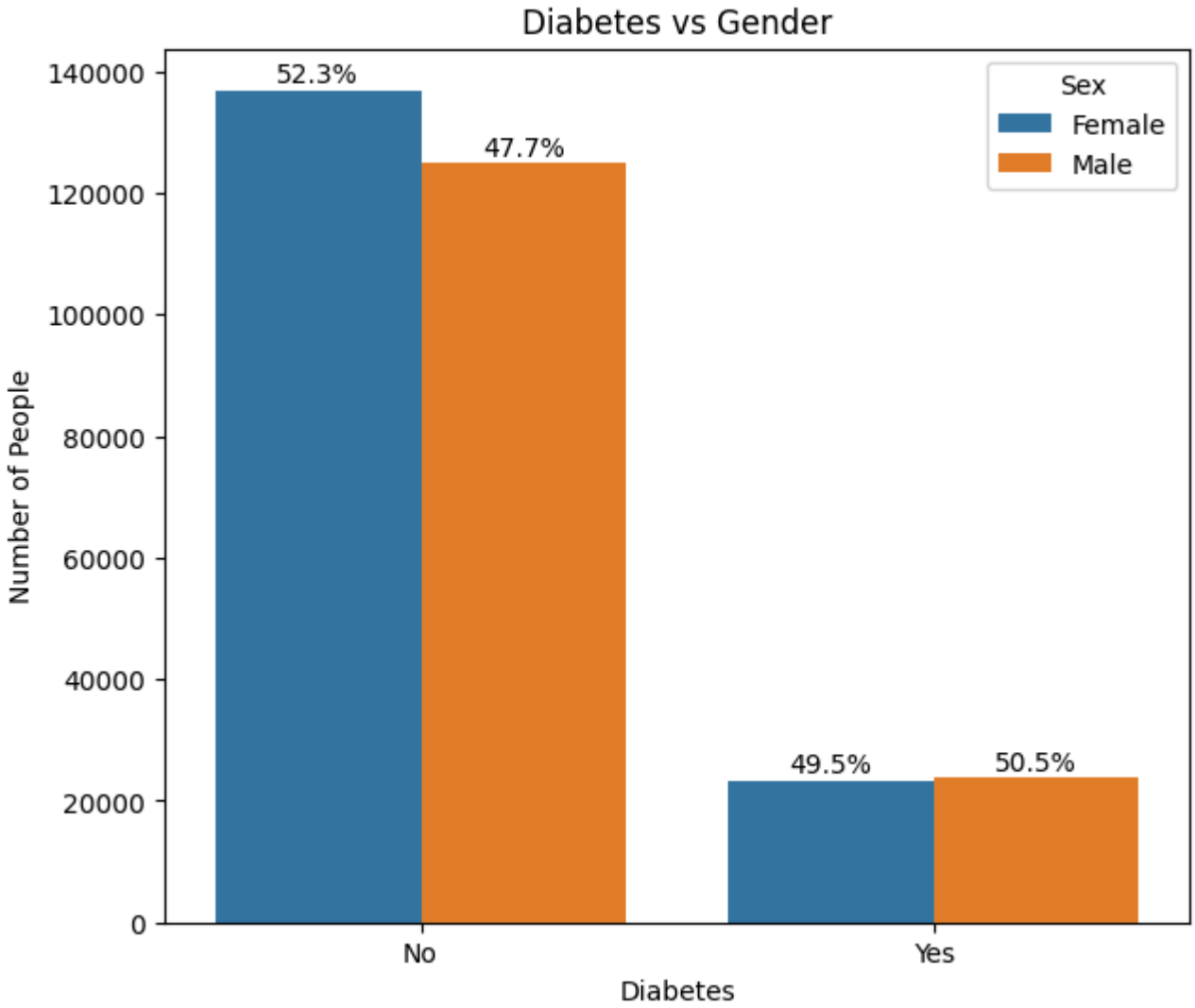
H_0 : Diabetes is not more common in men than in women $(p_1 - p_2 \leq 0)$

H_1 : Diabetes is more common in men than in women $(p_1 - p_2 > 0)$

This is a right-tailed test.

In [99]:
```
dfo.replace(to_replace="No, pre-diabetes or borderline diabetes",
            value="Yes", inplace=True)
dfo.replace(to_replace="Yes, but female told only during pregnancy",
            value="No", inplace=True)
column = 'Diabetes'
target = 'Sex'

fig,ax = plt.subplots(figsize = (7,6))

#color = 'Set2'

#palette_color = sns.color_palette(color)

ax = sns.countplot(x = column, data=dfo, hue=target, order = dfo[column].value_counts().index)
ax.set_ylabel('Count')

offset = dfo[column].value_counts().max() * 0.005

list_bars = dfo.groupby([column,target])[column].agg(['count']).unstack().fillna(0).values

patches = ax.patches
bars_pos = 0

for i in range(dfo[target].nunique()):
    for j in range(dfo[column].nunique()):
        list_bars_col = list_bars[j]
```

```
        total_sum = list_bars_col.sum()
        value = list_bars_col[i]

        percentage = value / total_sum

        if percentage == 0:
            bars_pos += 1
            continue
        else:
            x = patches[bars_pos].get_x() + patches[j].get_width()/2
            y = patches[bars_pos].get_height() + offset
            ax.annotate('{:.1f}%'.format(percentage*100), (x, y), ha='center')
            bars_pos += 1
plt.ylabel("Number of People")
plt.title("Diabetes vs Gender")
plt.show()
```



In [100…

```
df_female = dfo.loc[dfo["Sex"]=="Female", :]
df_male = dfo.loc[dfo["Sex"]=="Male", :]

male_diab = df_male["Diabetes"]
female_diab = df_female["Diabetes"]
male_non_diab = male_diab.value_counts().iloc[0]
female_non_diab = female_diab.value_counts().iloc[0]
male_diab_num = male_diab.value_counts().iloc[1]
female_diab_num = female_diab.value_counts().iloc[1]
n_male = len(male_diab)
```

```
n_female = len(female_diab)
p1 = male_diab_num/n_male       # (sample) proportion of diabetic men
p2 = female_diab_num/n_female


print("Number of diabetic men: {}, and number of men who are not diabetic: {}".format(male_diab
print("Number of depressed women: {}, and number of women who are not diabetic: {}".format(fema

print("(Sample) Proportion of diabetic men (p1):", p1)
print("(Sample) Proportion of diabetic women (p2):", p2)
```

```
Number of diabetic men: 23765, and number of men who are not diabetic: 124893
Number of depressed women: 23302, and number of women who are not diabetic: 136894
(Sample) Proportion of diabetic men (p1): 0.1598635794911811
(Sample) Proportion of diabetic women (p2): 0.14545931234238058
```

In [101…

```
temp = math.sqrt(((p1*(1-p1))/n_male)+((p2*(1-p2))/n_female))
z_test = (p1-p2)/temp
print("Test statistic:", z_test)
```

```
Test statistic: 11.115136439325367
```

Rejection Region Approach

We reject H_0 if $z_{test} \geq z_\alpha$ at level of significance α.

In [102…

```
z_crit = stats.norm.ppf(q=1-0.05)
z_crit
```

Out[102…

```
1.6448536269514722
```

Clearly, $z_{test} \geq z_\alpha$, so we reject H_0.

p-Value Approach

p value = $P(Z > Z_{test})$. We reject H_0 if $p < \alpha$.

In [103…

```
p = 1 - stats.norm.cdf(z_test)
print("p-value:", p)
```

```
p-value: 0.0
```

We reject the null hypothesis.

3. Hypothesis Testing on Categorical Variables

We now test some hypotheses regarding the association of factors to heart diseases. To this end, we randomly undersample the dataset first to balance the class of "heart disease". There is a significant imbalance in this class with only 9% of the people having heart disease, which might cause a bias towards the majority class.

3.1 Importing the Randomly Undersampled Dataset

In [104…]
```
df = pd.read_csv('/kaggle/input/undersampled-data/Undersampled_Data.csv')
df.head()
```

Out[104…]

	ID	General_Health	Checkup	Exercise	Heart_Disease	Skin_Cancer	Other_Cancer	Depression	D
0	147631	Excellent	Within the past 5 years	Yes	No	No	No	No	
1	159300	Fair	Within the past year	No	No	No	No	No	
2	227474	Good	Within the past 2 years	Yes	No	No	No	No	
3	114245	Very Good	Within the past 2 years	Yes	No	No	No	Yes	
4	74428	Fair	Within the past year	No	No	No	No	Yes	

In [105…]
```
df.shape
```

Out[105…]
```
(49942, 20)
```

3.2 Is Diabetes related to Heart Disease?

In [106…]
```
target = 'Heart_Disease'
```

In [107…]
```
df.replace(to_replace="No, pre-diabetes or borderline diabetes",
           value="Yes", inplace=True)
df.replace(to_replace="Yes, but female told only during pregnancy",
           value="No", inplace=True)
```

In [108…]
```
from scipy.stats import chi2_contingency

table = pd.crosstab(df["Diabetes"], df['Heart_Disease'])
table
```

Out[108…]

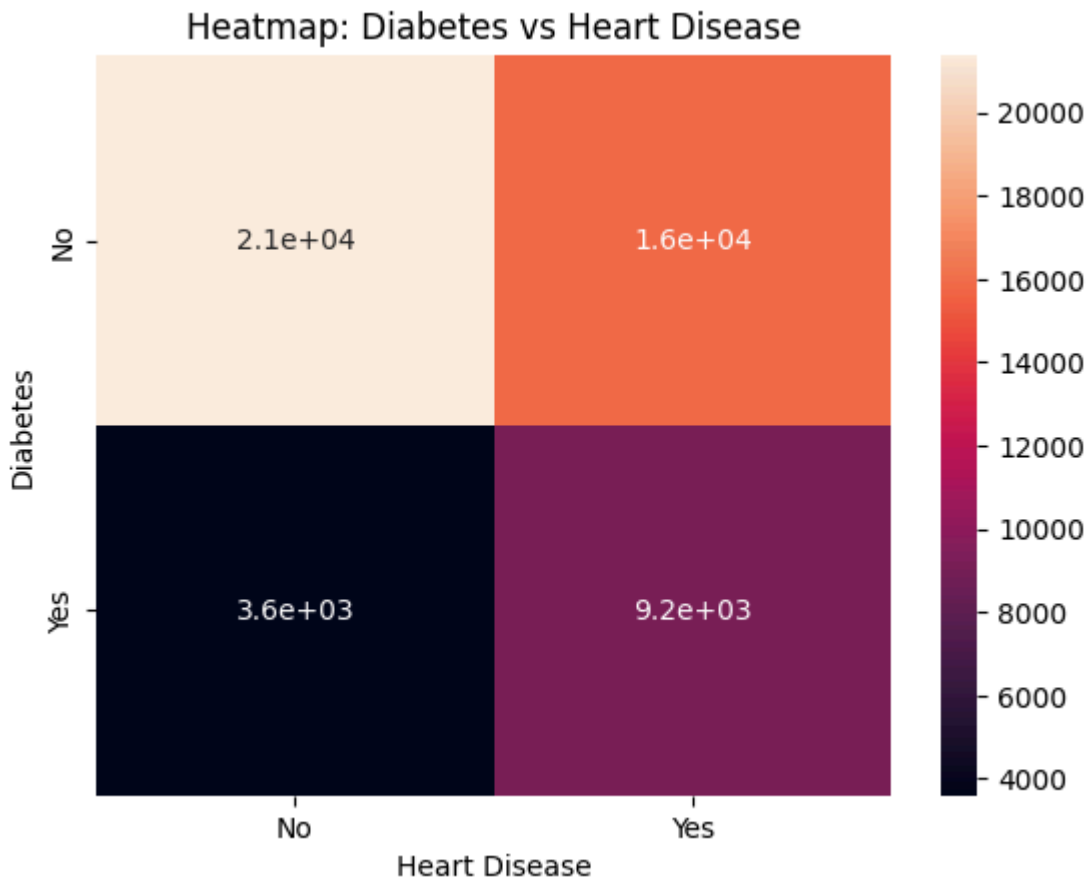
Heart_Disease	No	Yes
Diabetes		
No	21368	15801
Yes	3603	9170

In [109…]
```
test_result = chi2_contingency(table)
print("Test statistic: {}\np-value: {}".format(test_result[0], test_result[1]))
```

```
Test statistic: 3258.95636185264
p-value: 0.0
```

In [110…
```
sns.heatmap(table, annot=True)
plt.xlabel("Heart Disease")
plt.title("Heatmap: Diabetes vs Heart Disease")
```

Out[110… `Text(0.5, 1.0, 'Heatmap: Diabetes vs Heart Disease')`

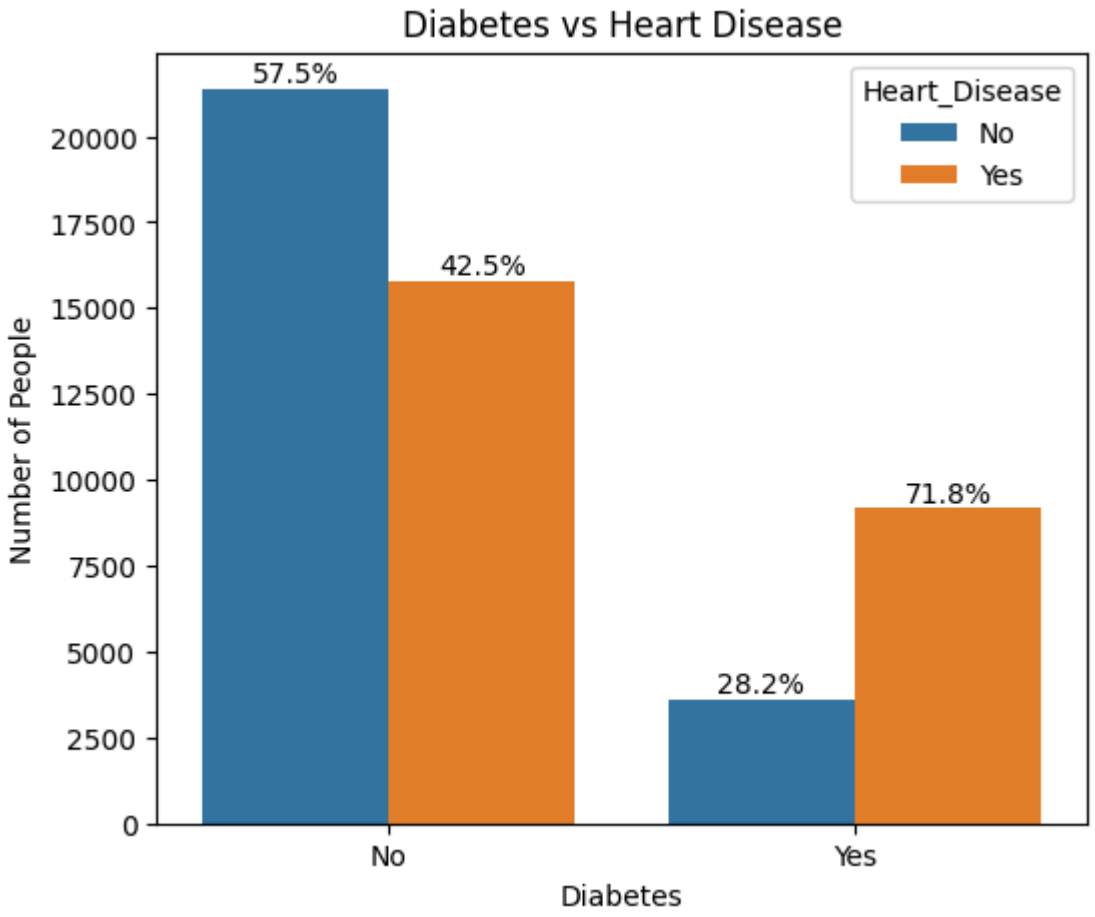


- As $\chi^2_{test} > \chi^2_{critical}$, we reject the null hypothesis H_0.
- Also, $p \approx 0 < 0.05 = \alpha$, so the p-value approach also leads us to rejecting H_0.

Hence, diabetes is related with heart diseases.

In [111…
```
#Diabetes vs Heart Disease

column = 'Diabetes'

fig,ax = plt.subplots(figsize = (6,5))

#color = 'Set2'

#palette_color = sns.color_palette(color)

ax = sns.countplot(x = column, data=df, hue=target, order = df[column].value_counts().index)
ax.set_ylabel('Count')

offset = df[column].value_counts().max() * 0.005

list_bars = df.groupby([column,target])[column].agg(['count']).unstack().fillna(0).values
```

```
patches = ax.patches
bars_pos = 0

for i in range(df[target].nunique()):
    for j in range(df[column].nunique()):
        list_bars_col = list_bars[j]
        total_sum = list_bars_col.sum()
        value = list_bars_col[i]

        percentage = value / total_sum

        if percentage == 0:
            bars_pos += 1
            continue
        else:
            x = patches[bars_pos].get_x() + patches[j].get_width()/2
            y = patches[bars_pos].get_height() + offset
            ax.annotate('{:.1f}%'.format(percentage*100), (x, y), ha='center')
            bars_pos += 1

plt.ylabel("Number of People")
plt.title("Diabetes vs Heart Disease")
plt.show()
```



3.3 Is Arthritis related to Heart Disease?

In [112…

```
table = pd.crosstab(df["Arthritis"], df['Heart_Disease'])
table
```

Out[112…

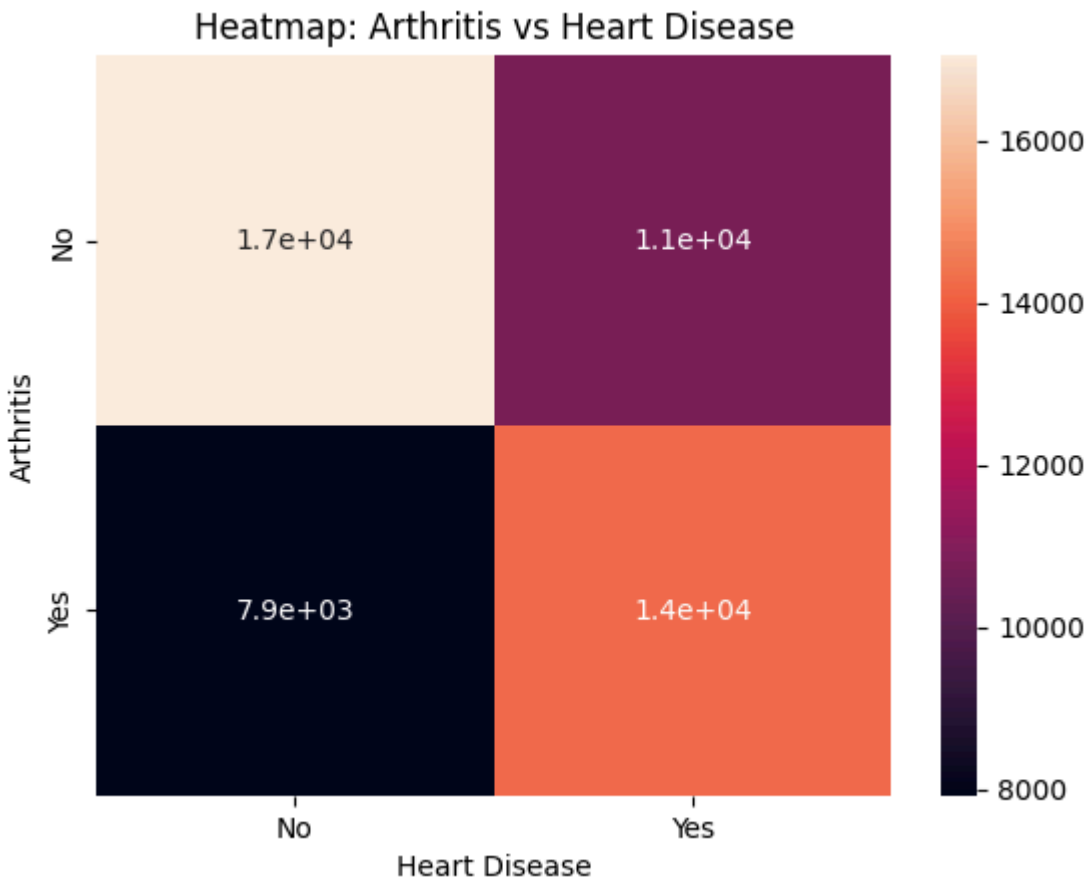
Heart_Disease	No	Yes
Arthritis		
No	17048	10719
Yes	7923	14252

In [113…

```
sns.heatmap(table, annot=True)
plt.xlabel("Heart Disease")
plt.title("Heatmap: Arthritis vs Heart Disease")
```

Out[113…

```
Text(0.5, 1.0, 'Heatmap: Arthritis vs Heart Disease')
```



In [114…

```
test_result = chi2_contingency(table)
print("Test statistic: {}\np-value: {}".format(test_result[0], test_result[1]))
```

```
Test statistic: 3247.9271719144276
p-value: 0.0
```

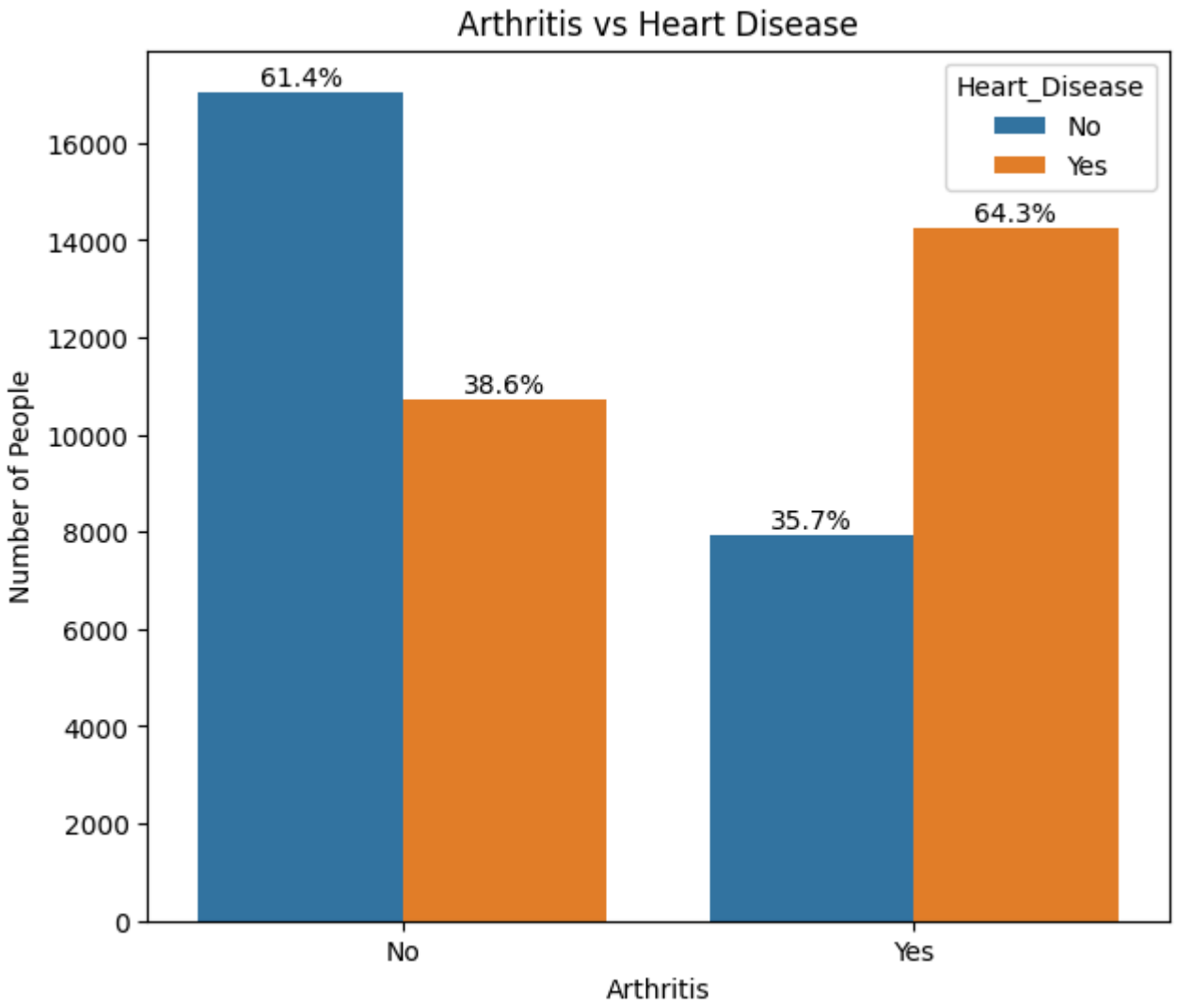
- As $\chi^2_{test} > \chi^2_{critical}$, we reject the null hypothesis H_0.
- $p \approx 0 < 0.05 = \alpha$, so we reject H_0 through the p-value approach as well.

Thus, arthritis is associated with heart diseases.

In [115…

```
column = 'Arthritis'

fig,ax = plt.subplots(figsize = (7,6))

#color = 'Set2'
```

```
#palette_color = sns.color_palette(color)

ax = sns.countplot(x = column, data=df, hue=target, order = df[column].value_counts().index)
ax.set_ylabel('Count')

offset = df[column].value_counts().max() * 0.005

list_bars = df.groupby([column,target])[column].agg(['count']).unstack().fillna(0).values

patches = ax.patches
bars_pos = 0

for i in range(df[target].nunique()):
    for j in range(df[column].nunique()):
        list_bars_col = list_bars[j]
        total_sum = list_bars_col.sum()
        value = list_bars_col[i]

        percentage = value / total_sum

        if percentage == 0:
            bars_pos += 1
            continue
        else:
            x = patches[bars_pos].get_x() + patches[j].get_width()/2
            y = patches[bars_pos].get_height() + offset
            ax.annotate('{:.1f}%'.format(percentage*100), (x, y), ha='center')
            bars_pos += 1
plt.ylabel("Number of People")
plt.title("Arthritis vs Heart Disease")
plt.show()
```



3.4 Is Age Associated with Heart Disease?

In [116…

```
table = pd.crosstab(df["Age_Category"], df['Heart_Disease'])
table
```

Out[116…

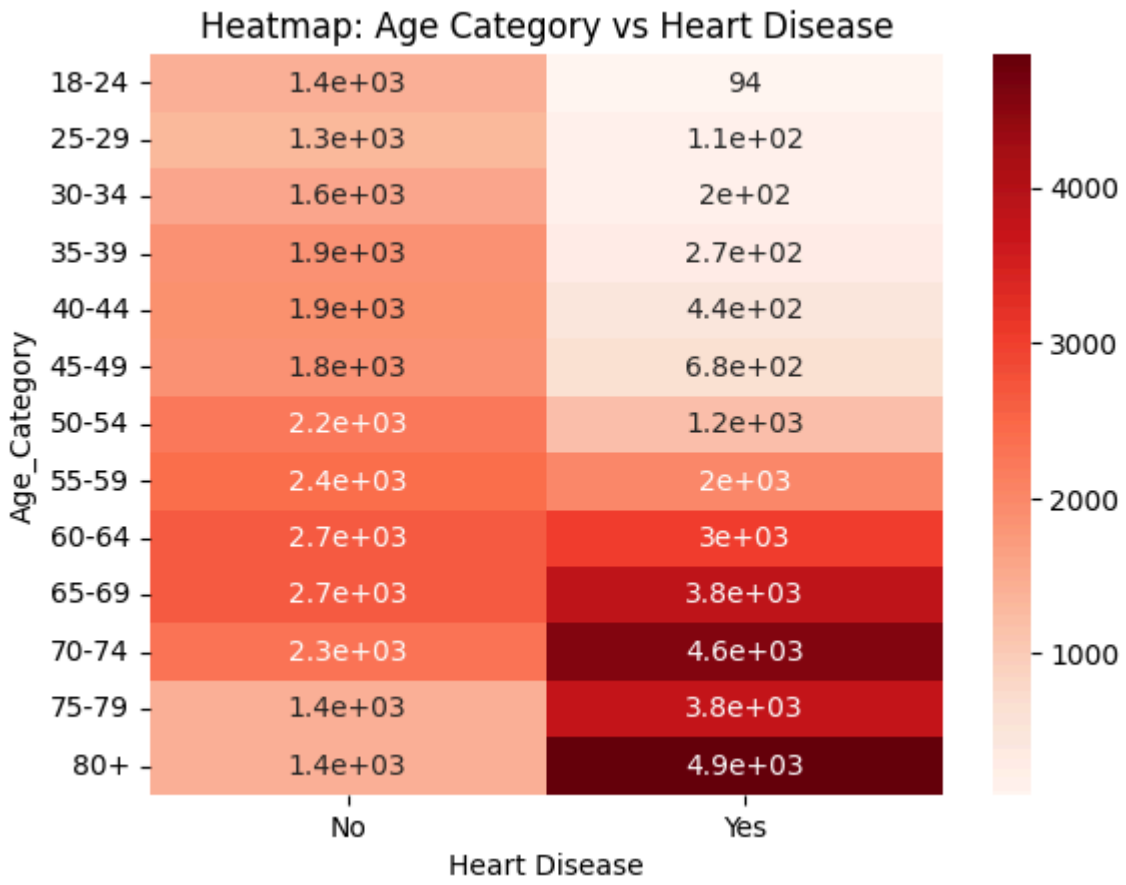
Heart_Disease	No	Yes
Age_Category		
18-24	1414	94
25-29	1310	113
30-34	1550	201
35-39	1859	274
40-44	1895	435
45-49	1848	678
50-54	2246	1181
55-59	2415	1991
60-64	2653	3012
65-69	2653	3823
70-74	2292	4561
75-79	1429	3752
80+	1407	4856

In [117…

```
sns.heatmap(table, annot=True, cmap='Reds')
plt.xlabel("Heart Disease")
plt.title("Heatmap: Age Category vs Heart Disease")
```

Out[117…

```
Text(0.5, 1.0, 'Heatmap: Age Category vs Heart Disease')
```



In [118…
```
test_result = chi2_contingency(table)
print("Test statistic: {}\np-value: {}".format(test_result[0], test_result[1]))
```

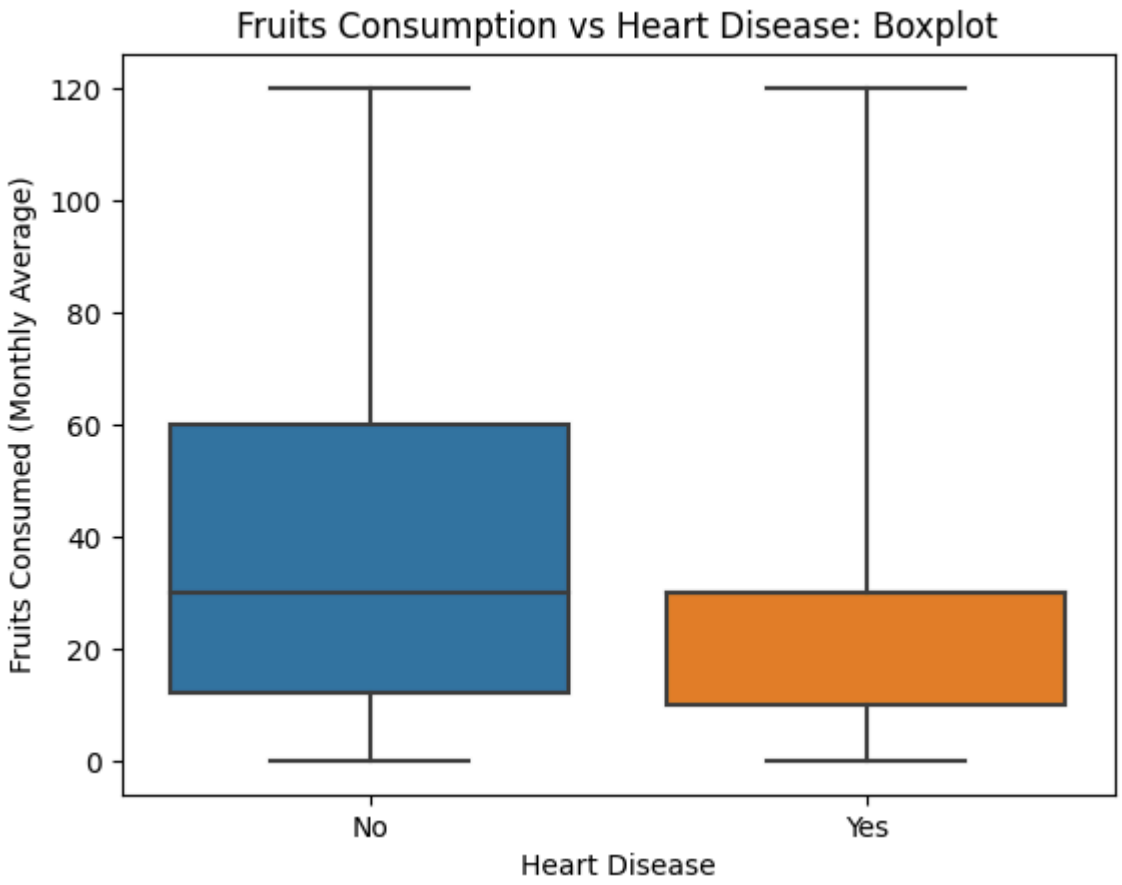
```
Test statistic: 10134.250600009203
p-value: 0.0
```

- As $\chi^2_{test} > \chi^2_{critical}$, we reject the null hypothesis H_0.
- $p \approx 0 < 0.05 = \alpha$, so we reject H_0 through the p-value approach as well.

Hence, we conclude that we have enough statistical evidence to claim that age is related to heart diseases.

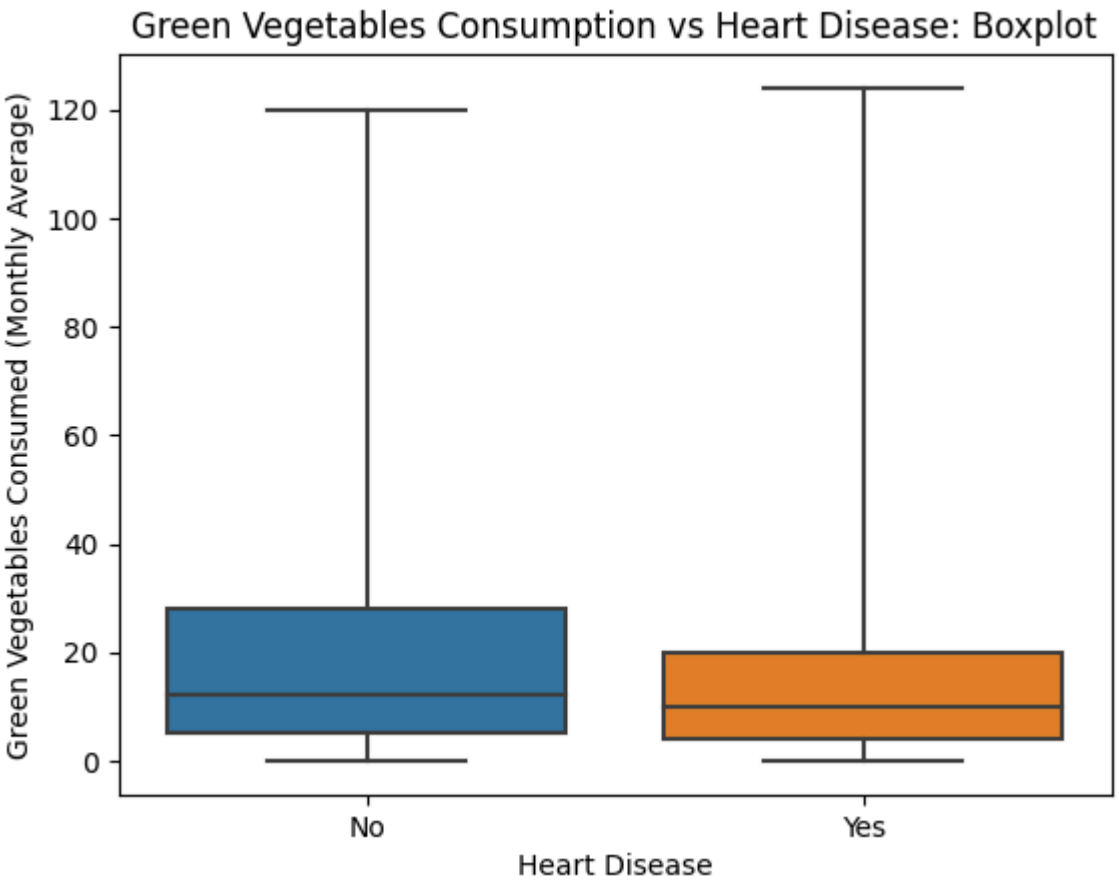
4. Fruits & Vegetables Consumption vs Heart Disease

In [119…
```
sns.boxplot(x=df["Heart_Disease"], y=df["Fruit_Consumption"], whis=(0,100), orient='v')
plt.title("Fruits Consumption vs Heart Disease: Boxplot")
plt.xlabel("Heart Disease")
plt.ylabel("Fruits Consumed (Monthly Average)")
plt.show()
```



In [120…

```
sns.boxplot(data=df, x=df["Heart_Disease"], y=df["Green_Vegetables_Consumption"], whis=(0,100),
plt.title("Green Vegetables Consumption vs Heart Disease: Boxplot")
plt.xlabel("Heart Disease")
plt.ylabel("Green Vegetables Consumed (Monthly Average)")
plt.show()
```



Author: Rajdeep Pathak

Department of Mathematics and Computing, IIT Hyderabad

Date: 03/05/2024