1.

a)

import pandas as pd

import numpy as np

import random

import matplotlib.pyplot as plt

# load the diabetes data

diabetes = pd.read\_csv('diabetes.csv')

# set a seed for reproducibility

random.seed(123)

# take a random sample of 25 observations

sample = diabetes.sample(n=25)

# calculate the mean and highest glucose values of the sample

sample\_mean = sample['Glucose'].mean()

sample\_highest = sample['Glucose'].max()

# calculate the population mean and highest glucose values

population\_mean = diabetes['Glucose'].mean()

population\_highest = diabetes['Glucose'].max()

# plot the comparison between the sample and population means and highest values

fig, axs = plt.subplots(1, 2, figsize=(12, 5))

axs[0].bar(['Sample Mean', 'Population Mean'], [sample\_mean, population\_mean])

axs[0].set\_title('Comparison of Sample and Population Means')

axs[1].bar(['Sample Highest', 'Population Highest'], [sample\_highest, population\_highest])

axs[1].set\_title('Comparison of Sample and Population Highest Values')

plt.show()

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

Chart, bar chart

Description automatically generated

b)

# find the 98th percentile of BMI for the sample and population

sample\_98th = np.percentile(sample['BMI'], 98)

population\_98th = np.percentile(diabetes['BMI'], 98)

# plot the comparison between the sample and population 98th percentiles

fig, axs = plt.subplots(figsize=(8, 5))

axs.bar(['Sample 98th Percentile', 'Population 98th Percentile'], [sample\_98th, population\_98th])

axs.set\_title('Comparison of Sample and Population 98th Percentiles of BMI')

plt.show()

Sample percentile: 40.248

Population percentile: 47.52599999999996

Chart, bar chart

Description automatically generated

c)

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

# load the diabetes dataset

df = pd.read\_csv('diabetes.csv')

# extract the BloodPressure variable

bp = df['BloodPressure']

# population statistics

pop\_mean = bp.mean()

pop\_std = bp.std()

pop\_pct = np.percentile(bp, 95)

# bootstrap sampling

np.random.seed(123)

bootstrap\_means = []

bootstrap\_stds = []

bootstrap\_pcts = []

for i in range(500):

bootstrap\_sample = np.random.choice(bp, size=150, replace=True)

bootstrap\_means.append(bootstrap\_sample.mean())

bootstrap\_stds.append(bootstrap\_sample.std())

bootstrap\_pcts.append(np.percentile(bootstrap\_sample, 95))

# plot the distributions

fig, axs = plt.subplots(3, figsize=(8, 12))

axs[0].hist(bp, bins=20, alpha=0.5)

axs[0].set\_title('Population Distribution')

axs[0].axvline(pop\_mean, color='red', label='Population Mean')

axs[0].legend()

axs[1].hist(bootstrap\_means, bins=20, alpha=0.5)

axs[1].set\_title('Bootstrap Mean Distribution')

axs[1].axvline(np.mean(bootstrap\_means), color='red', label='Bootstrap Mean')

axs[1].legend()

axs[2].hist(bootstrap\_stds, bins=20, alpha=0.5)

axs[2].set\_title('Bootstrap Standard Deviation Distribution')

axs[2].axvline(np.mean(bootstrap\_stds), color='red', label='Bootstrap Standard Deviation')

axs[2].legend()

plt.show()

# print the statistics

print('Population Mean: {:.2f}'.format(pop\_mean))

print('Bootstrap Mean: {:.2f}'.format(np.mean(bootstrap\_means)))

print('Population Standard Deviation: {:.2f}'.format(pop\_std))

print('Bootstrap Standard Deviation: {:.2f}'.format(np.mean(bootstrap\_stds)))

print('Population 95th Percentile: {:.2f}'.format(pop\_pct))

print('Bootstrap 95th Percentile: {:.2f}'.format(np.mean(bootstrap\_pcts)))

Timeline

Description automatically generated

Text

Description automatically generated