Question 1

In [2]:
```
# Generate population from Poisson Distribution
import numpy as np
import pandas as pd
# Set random seed for reproducibilty
np.random.seed(123)
# Generate populaiton of size 1000 from poisson (λ=20)
population = np.random.poisson(lam=20, size=1000)

print("First 10 values of the population sample:", population[:10])
print("Populatio mean:", np.mean(population))
print("Population std deviation:",np.std(population))
```

```
First 10 values of the population sample: [23 16 23 20 18 19 19 15 20 22]
Populatio mean: 19.987
Population std deviation: 4.54299801892979
```

In [5]:
```
print("First 15 elements:\n", population[:15],
      "\n\nLast 15 elements:\n", population[-15:])
```

```
First 15 elements:
 [23 16 23 20 18 19 19 15 20 22 23 23 18 17 13]

Last 15 elements:
 [20 24 25 21 34 22 19 23 26 19 22 19 20 24 23]
```

In [17]:
```
# Prepare Bootstrap sample of size 10
sample10=np.random.choice(population, size=10, replace=True)
sample10
```

Out[17]:
```
array([23, 21, 21, 15, 20, 23, 28, 15, 20, 19], dtype=int32)
```

In [18]:
```
# Bootstrap the mean
n_bootstrap=30
boot_means_10=[]
for _ in range(n_bootstrap):
  boot_sample=np.random.choice(sample10, size=10, replace=True)
  boot_means_10.append(np.mean(boot_sample))
boot_means_10
```

Out[18]:
```
[np.float64(18.3),
 np.float64(19.7),
 np.float64(21.9),
 np.float64(20.5),
 np.float64(19.6),
 np.float64(21.3),
 np.float64(22.7),
 np.float64(21.7),
 np.float64(20.3),
 np.float64(20.0),
 np.float64(21.3),
 np.float64(22.8),
 np.float64(19.4),
 np.float64(22.3),
 np.float64(21.6),
 np.float64(21.5),
 np.float64(19.3),
 np.float64(19.7),
 np.float64(19.2),
 np.float64(20.2),
 np.float64(19.9),
 np.float64(19.2),
 np.float64(19.7),
 np.float64(20.5),
 np.float64(22.3),
 np.float64(22.0),
 np.float64(22.1),
 np.float64(21.6),
 np.float64(18.9),
 np.float64(20.4)]
```

In [19]:
```
# Mean estimate from bootstrap
mean_boot_10=np.mean(boot_means_10)
print(f"{mean_boot_10:.2f}")
```

```
20.66
```

In [20]:
```
# Confidence Interval
ci_95_10=np.percentile(boot_means_10, [2.5, 97.5])
ci_99_10=np.percentile(boot_means_10, [0.5, 99.5])
print("\nSample size 10:")
print(f"Bootstrap Mean estimate: {mean_boot_10:.2f}")
print(f"95% CI: {ci_95_10}")
print(f"99% CI: {ci_95_10}")
```

```
Sample size 10:
Bootstrap Mean estimate: 20.66
95% CI: [18.735  22.7275]
99% CI: [18.735  22.7275]
```

In [21]:
```
# Draw bootstrap sample of size 100
sample_100=np.random.choice(population, size=100, replace=True)
sample_100

# Bootstrap the mean
boot_means_100=[]
for _ in range(n_bootstrap):
```

```python
  boot_sample=np.random.choice(sample_100, size=100, replace=True)
  boot_means_100.append(np.mean(boot_sample))

# Mean estimate from bootstrap
mean_boot_100=np.mean(boot_means_100)

# Confidence Interval
ci_95_100=np.percentile(boot_means_100, [2.5, 97.5])
ci_99_100=np.percentile(boot_means_100, [0.5, 99.5])
print("\nSample size 100:")
print(f"Bootstrap Mean estimate: {mean_boot_100:.2f}")
print(f"95% CI: {ci_95_100}")
print(f"99% CI: {ci_99_100}")
```

```
Sample size 100:
Bootstrap Mean estimate: 20.74
95% CI: [19.7845 21.5765]
99% CI: [19.5409 21.6113]
```

In [22]:
```python
# Data from the bootstrap analysis summary
data = {
    'Statistic': [
        'Bootstrap Mean Estimate',
        '95% Confidence Interval',
        '99% Confidence Interval'
    ],
    'Sample Size 10': [
         mean_boot_10,
        ci_95_10,
        ci_99_10
    ],
    'Sample Size 100': [
        mean_boot_100,
        ci_95_100,
        ci_99_100
    ]
}
# Create a pandas DataFrame from the data
df = pd.DataFrame(data)
# Set the 'Statistic' column as the index for better table representation
df.set_index('Statistic', inplace=True)

# --- Display the DataFrame ---
# The print output will be a formatted table.
print("Bootstrap Analysis Summary")
print("-" * 30)
print(df)
```

```
Bootstrap Analysis Summary
------------------------------
                            Sample Size 10               Sample Size 100
Statistic
Bootstrap Mean Estimate          20.663333                        20.737
95% Confidence Interval  [18.735, 22.7275]  [19.784499999999998, 21.5765]
99% Confidence Interval  [18.387, 22.7855]              [19.5409, 21.6113]
```

Question 2

In [23]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
```

In [24]:
```
# CGPA data
cgpa_data = np.array([
    3.6, 5.2, 4.5, 8.4, 6.0, 4.5, 5.7, 6.5, 7.4, 5.7,
    6.8, 7.6, 4.0, 6.8, 7.9, 6.1, 2.8, 7.9, 3.3, 4.4,
    4.9, 3.3, 5.9, 3.2, 8.0, 5.9, 7.5, 6.6, 4.4, 7.5,
    7.4, 6.2, 3.0, 4.1, 3.0, 7.1, 5.8, 5.2, 8.4, 3.1
])
```

In [25]:
```
# a) Population Stats
population_mean = np.mean(cgpa_data)
population_median = np.median(cgpa_data)
population_std = np.std(cgpa_data, ddof=0)  # Population standard deviation
population_range = np.max(cgpa_data) - np.min(cgpa_data)
print("a) Population Statistics:")
print(f"Mean: {population_mean:.2f}")
print(f"Median: {population_median:.2f}")
print(f"Standard Deviation: {population_std:.2f}")
print(f"Range: {population_range:.2f}\n")
```

```
a) Population Statistics:
Mean: 5.64
Median: 5.85
Standard Deviation: 1.71
Range: 5.60
```

In [26]:
```
# b) Bootstrap sample of size 20 (with replacement)
bootstrap_sample_20 = np.random.choice(cgpa_data, size=20, replace=True)
sample_mean_20 = np.mean(bootstrap_sample_20)
sample_median_20 = np.median(bootstrap_sample_20)
sample_std_20 = np.std(bootstrap_sample_20, ddof=1)  # Sample standard deviation

print("b) Bootstrap Sample of Size 20:")
print(f"Sample Mean: {sample_mean_20:.2f}")
print(f"Sample Median: {sample_median_20:.2f}")
print(f"Sample Std Dev: {sample_std_20:.2f}\n")
```

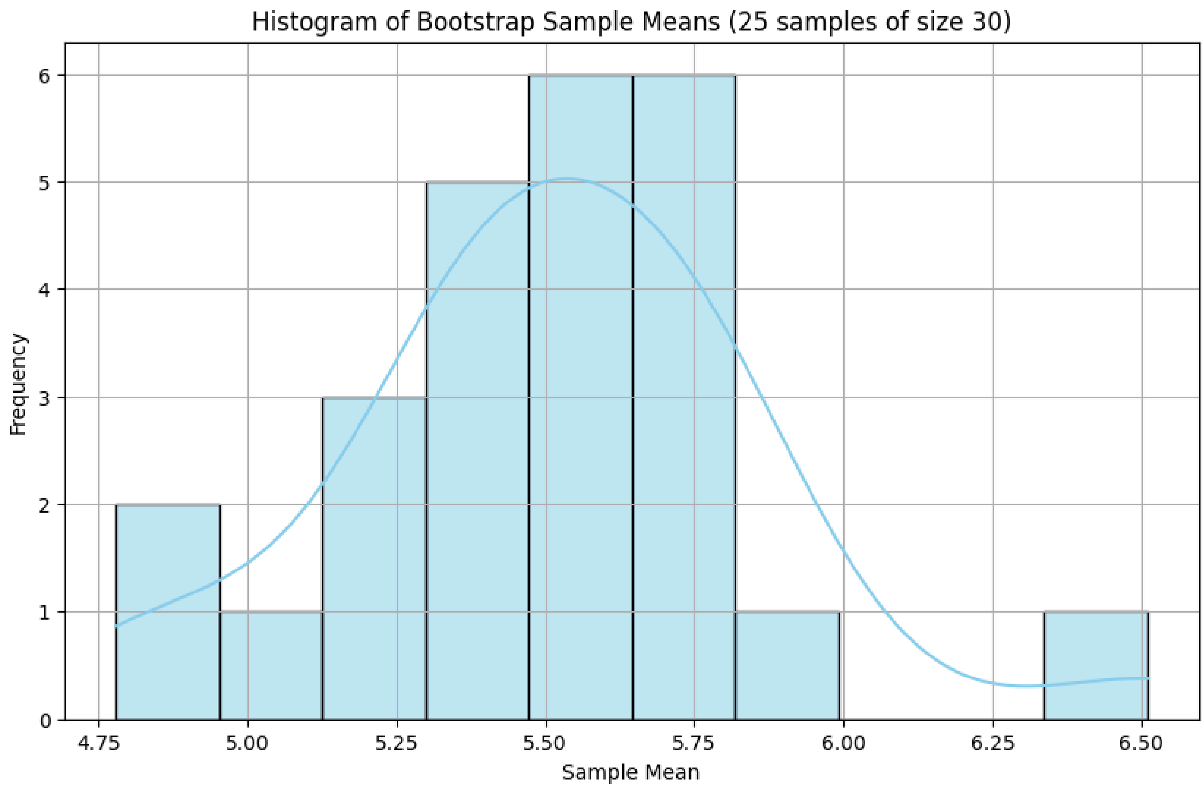
```
b) Bootstrap Sample of Size 20:
Sample Mean: 4.71
Sample Median: 4.45
Sample Std Dev: 1.41
```

In [27]:
```
# c) Bootstrapping: 25 samples of size 30
bootstrap_means = []
num_samples = 25
sample_size = 30

for _ in range(num_samples):
    sample = np.random.choice(cgpa_data, size=sample_size, replace=True)
```

```python
    mean = np.mean(sample)
    bootstrap_means.append(mean)

# Plot histogram of bootstrap means
plt.figure(figsize=(10, 6))
sns.histplot(bootstrap_means, bins=10, kde=True, color='skyblue', edgecolor='black'
plt.title('Histogram of Bootstrap Sample Means (25 samples of size 30)')
plt.xlabel('Sample Mean')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



In [28]:
```python
# d) Estimate 95% Confidence Interval using bootstrap sample means
lower_bound = np.percentile(bootstrap_means, 2.5)
upper_bound = np.percentile(bootstrap_means, 97.5)
print("d) 95% Confidence Interval for Mean CGPA (using bootstrap):")
print(f"95% CI: ({lower_bound:.2f}, {upper_bound:.2f})")
```

```
d) 95% Confidence Interval for Mean CGPA (using bootstrap):
95% CI: (4.84, 6.18)
```

In [29]:
```python
# Data from the bootstrap analysis summary
data = {
    'Statistic': ['Bootstrap Sample (n=20)', '95% CI for Mean'],
    'Mean': [sample_mean_20, np.mean(bootstrap_means)],
    'Median': [sample_median_20, np.nan],
    'Std Dev': [sample_std_20, np.nan],
    'Lower Bound': [np.nan, lower_bound],
    'Upper Bound': [np.nan, upper_bound]
}
df = pd.DataFrame(data)
df.set_index('Statistic', inplace=True)
```

```
print("\nBootstrap Analysis Summary")
print("-" * 30)
print(df.round(2))
```

```
Bootstrap Analysis Summary
------------------------------
                          Mean  Median  Std Dev  Lower Bound  Upper Bound
Statistic
Bootstrap Sample (n=20)   4.70    4.45     1.41          NaN          NaN
95% CI for Mean           5.52     NaN      NaN         4.84         6.18
```

Question 3

In [30]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression

# Original data
experience = np.array([1,2,3,4,5,6,7,8,9,10])
salary = np.array([32,33,35,38,37,40,43,45,47,49])

# Combine into a DataFrame for easy bootstrap sampling
data3 = pd.DataFrame({'Experience': experience, 'Salary': salary})
data3
```

Out[30]:

	Experience	Salary
0	1	32
1	2	33
2	3	35
3	4	38
4	5	37
5	6	40
6	7	43
7	8	45
8	9	47
9	10	49

In [31]:
```
# Function to fit regression and return slope
def fit_regression(X, y):
    model = LinearRegression()
    model.fit(X.reshape(-1, 1), y)
    return model.coef_[0], model.intercept_
```

In [32]:

```python
# a & b) Create 3 bootstrap samples and fit regression models
slopes = []
intercepts = []
```

In [33]:

```python
# b) Create 3 bootstrap samples and fit regression models
print("b) Regression Coefficients for 3 Bootstrap Samples:")

for i in range(1, 4):
    sample = data3.sample(n=10, replace=True)  # bootstrap sample
    X_sample = sample['Experience'].values
    y_sample = sample['Salary'].values
    slope, intercept = fit_regression(X_sample, y_sample)
    slopes.append(slope)
    intercepts.append(intercept)
b_data = {
    "Sample": [f"Sample {i}" for i in range(1, len(slopes)+1)],
    "Slope": [round(s, 4) for s in slopes],
    "Intercept": [round(it, 4) for it in intercepts]
}
b_df = pd.DataFrame(b_data).set_index("Sample")

print("\n(b) Regression Coefficients (Table):")
print("-" * 40)
print(b_df)

# c) Interpretation
pd.set_option("display.max_colwidth", None)
print("\n(c) Interpretation:")

c_data = {
    "Sample": [f"Sample {i}" for i in range(1, len(slopes)+1)],
    "Interpretation": [
        f"For each additional year of experience, salary increases by approx. {s:.2
        for s in slopes
    ]
}
c_df = pd.DataFrame(c_data).set_index("Sample")

print("-" * 40)
print(c_df)
```

```
b) Regression Coefficients for 3 Bootstrap Samples:

(b) Regression Coefficients (Table):
----------------------------------------
           Slope  Intercept
Sample
Sample 1  1.9538    28.7172
Sample 2  1.8995    29.5825
Sample 3  1.8492    29.8827

(c) Interpretation:
----------------------------------------
                                                                                Interpretation
Sample
Sample 1  For each additional year of experience, salary increases by approx. 1.95k.
Sample 2  For each additional year of experience, salary increases by approx. 1.90k.
Sample 3  For each additional year of experience, salary increases by approx. 1.85k.
```