## Random Sampling

In [2]:
```
import random

population = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sample = random.sample(population, k=5)
print(sample)
```

```
[8, 4, 10, 9, 5]
```

In [3]:
```
import pandas as pd
import numpy as np

# Creating a sample DataFrame
data = {
    'EmployeeID': range(1, 101),
    'Name': [f'Employee {i}' for i in range(1, 101)],
    'Age': np.random.randint(22, 60, size=100),
    'Salary': np.random.randint(40000, 100000, size=100),
}

employee_data = pd.DataFrame(data)

# Display the first few rows of the DataFrame
print(employee_data.head())
```

```
   EmployeeID        Name  Age  Salary
0           1  Employee 1   56   78320
1           2  Employee 2   57   91856
2           3  Employee 3   26   70602
3           4  Employee 4   44   90823
4           5  Employee 5   31   96499
```

## Simple Random Sampling (SRS):

In [5]:

```
# Simple Random Sampling (SRS) - Select 10 random employees
srs_sample = employee_data.sample(n=10, random_state=42)
print("Simple Random Sampling:")
print(srs_sample)
```

```
Simple Random Sampling:
    EmployeeID         Name  Age  Salary
83          84  Employee 84   45   59064
53          54  Employee 54   26   40953
70          71  Employee 71   47   82590
45          46  Employee 46   54   42933
44          45  Employee 45   51   42116
39          40  Employee 40   43   97597
22          23  Employee 23   49   76958
80          81  Employee 81   48   50484
10          11  Employee 11   29   99270
0            1   Employee 1   56   78320
```

## Stratified Sampling:

In [6]:

```
# Create age groups
bins = [20, 30, 40, 50, 60]
labels = ['20-30', '31-40', '41-50', '51-60']
employee_data['AgeGroup'] = pd.cut(employee_data['Age'], bins=bins, labels=

# Stratified Sampling - Select 2 random employees from each age group
stratified_sample = employee_data.groupby('AgeGroup').apply(lambda x: x.sar
print("\nStratified Sampling:")
print(stratified_sample)
```

```
Stratified Sampling:
   EmployeeID         Name  Age  Salary AgeGroup
0          38  Employee 38   26   42432    20-30
1          82  Employee 82   24   72224    20-30
2           5   Employee 5   31   96499    31-40
3          59  Employee 59   40   48403    31-40
4          73  Employee 73   50   46497    41-50
5          43  Employee 43   48   72553    41-50
6           1   Employee 1   56   78320    51-60
7           2   Employee 2   57   91856    51-60
```

## Systematic Sampling:

In [7]:

```
# Systematic Sampling - Select every 10th employee starting from the 5th
systematic_sample = employee_data.iloc[4::10]
print("\nSystematic Sampling:")
print(systematic_sample)
```

```
Systematic Sampling:
    EmployeeID         Name  Age  Salary AgeGroup
4            5   Employee 5   31   96499    31-40
14          15  Employee 15   49   50501    41-50
24          25  Employee 25   46   72076    41-50
34          35  Employee 35   54   88073    51-60
44          45  Employee 45   51   42116    51-60
54          55  Employee 55   39   62206    31-40
64          65  Employee 65   44   82324    41-50
74          75  Employee 75   46   51780    41-50
84          85  Employee 85   31   97190    31-40
94          95  Employee 95   24   73173    20-30
```

## Cluster Sampling:

In [10]:

```
# Create two clusters (departments)
departments = ['HR', 'Engineering']
employee_data['Department'] = np.random.choice(departments, size=100)

# Cluster Sampling - Select all employees from one randomly chosen departme
selected_department = np.random.choice(departments)
cluster_sample = employee_data[employee_data['Department'] == selected_depa
print(f"\nCluster Sampling (Department: {selected_department}):")
print(cluster_sample.head())
```

```
Cluster Sampling (Department: Engineering):
   EmployeeID        Name  Age  Salary AgeGroup   Department
0           1  Employee 1   56   78320    51-60  Engineering
3           4  Employee 4   44   90823    41-50  Engineering
4           5  Employee 5   31   96499    31-40  Engineering
6           7  Employee 7   27   67521    20-30  Engineering
7           8  Employee 8   47   80326    41-50  Engineering
```

## Random Sampling with Replacement:

In [9]:
```
# Random Sampling with Replacement - Select 10 employees with replacement
sample_with_replacement = employee_data.sample(n=10, replace=True, random_s
print("\nRandom Sampling with Replacement:")
print(sample_with_replacement)
```

```
Random Sampling with Replacement:
    EmployeeID         Name  Age  Salary AgeGroup   Department
51          52  Employee 52   26   74775    20-30           HR
92          93  Employee 93   58   67868    51-60  Engineering
14          15  Employee 15   49   50501    41-50           HR
71          72  Employee 72   32   56519    31-40           HR
60          61  Employee 61   58   80766    51-60           HR
20          21  Employee 21   44   85221    41-50           HR
82          83  Employee 83   47   60674    41-50  Engineering
86          87  Employee 87   26   45879    20-30           HR
74          75  Employee 75   46   51780    41-50           HR
74          75  Employee 75   46   51780    41-50           HR
```

**Sampling Distribution of the Sample Mean (Central Limit Theorem):**

**Population Distribution: Suppose we have a population with any distribution (not necessarily normal).**

**Sampling Distribution:** If we repeatedly take random samples of a fixed size from the population and calculate the mean of each sample, the distribution of those sample means will approach a normal distribution as the sample size increases, according to the Central Limit Theorem.

Example: Imagine measuring the heights of individuals from a diverse population and calculating the mean height for each sample. The sampling distribution of the sample means will tend to be approximately normal.

**Sampling Distribution of the Sample Proportion:**

**Population Distribution:** Suppose we have a population with two possible outcomes (e.g., success/failure).

**Sampling Distribution:** If we repeatedly take random samples of a fixed size from the population and calculate the proportion of successes in each sample, the distribution of those sample proportions will follow a normal distribution as the sample size increases, according to the Central Limit Theorem for Proportions.

Example: Conducting surveys to estimate the proportion of people who support a particular policy, and calculating the proportion of supporters in each sample.

### Bootstrap distribution

A bootstrap distribution is a sampling distribution of a statistic that is estimated by repeatedly resampling the original dataset with replacement. It allows us to approximate the sampling variability of a statistic without making strong assumptions about the population distribution. Here's an example of creating a bootstrap distribution for the mean of a dataset using Python:

In [20]:

```
import numpy as np

# Example dataset (you can replace this with your own data)
data = np.array([10, 15, 12, 18, 20, 14, 16, 11, 19, 13])

# Number of bootstrap samples to generate
num_samples = 1000

# Initialize an empty array to store the means from bootstrap samples
bootstrap_means = []

# Perform bootstrap resampling
for _ in range(num_samples):
    # Randomly sample with replacement from the original data
    bootstrap_sample = np.random.choice(data, size=len(data), replace=True)

    # Calculate the mean of the bootstrap sample
    bootstrap_mean = np.mean(bootstrap_sample)

    # Append the bootstrap mean to the list
    bootstrap_means.append(bootstrap_mean)

# Now, 'bootstrap_means' contains the means from 1000 bootstrap samples
print(bootstrap_mean)
```

```
14.5
```

In [21]:

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

# Create a histogram of bootstrap_means using seaborn
sns.histplot(bootstrap_means, bins=30, kde=True)
plt.xlabel('Sample Means')
plt.ylabel('Frequency')
plt.title('Bootstrap Distribution of the Mean')
plt.show()
```