Sampling Methods

What is Sampling Technique?

A sampling technique is a method used to select a representative subset (sample) from a larger population to make inferences about the entire population. It is used when it is not feasible or practical to collect data from every member of the population.

Why is Sampling Used?

Sampling is used for several reasons, including:

- **Cost-effectiveness:** Collecting data from a sample is generally less expensive and time-consuming than collecting data from the entire population.
- **Efficiency:** Analyzing a sample can be more efficient than analyzing the entire population, especially for large populations.
- **Practicality:** In some cases, it may be impossible to access or collect data from every member of the population.

Different Sampling Methods

Here are some common sampling methods:

1. Simple Random Sampling
2. Stratified Sampling
3. Cluster Sampling
4. Systematic Sampling

In [2]:
```
# Let's Load the Familiar Heart Disease Dataset
import pandas as pd
df = pd.read_csv("heart.csv")
df.head()
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       1025 non-null   int64  
 1   sex       1025 non-null   int64  
 2   cp        1025 non-null   int64  
 3   trestbps  1025 non-null   int64  
 4   chol      1025 non-null   int64  
 5   fbs       1025 non-null   int64  
 6   restecg   1025 non-null   int64  
 7   thalach   1025 non-null   int64  
 8   exang     1025 non-null   int64  
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64  
 11  ca        1025 non-null   int64  
 12  thal      1025 non-null   int64  
 13  target    1025 non-null   int64  
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

1. Simple Random Sampling

- **Definition:** In simple random sampling, every individual in the population has an equal chance of being selected for the sample.
- **Use Cases:** When the population is relatively homogeneous and there is no need to consider specific subgroups or characteristics.
- **Advantages:** Easy to implement and ensures that the sample is representative of the population.
- **Disadvantages:** May not be suitable for populations with diverse subgroups or when specific characteristics need to be represented.
- **Example (Heart Disease Dataset):** Randomly selecting 200 patients from the entire dataset to analyze their characteristics and predict heart disease risk.

In [4]:
```
# Randomly Selecting 200 patients
sample_df = df.sample(n=200, random_state=100)  # random_state ensures reproducibility

# Print the first few rows of the sample
print(sample_df.head())

print(sample_df.info())
# You can now perform further analysis or modeling on this sample
```

```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
960    52    0   2       136   196    0        0      169      0      0.1   
48     66    0   2       146   278    0        0      152      0      0.0   
886    61    1   0       120   260    0        1      140      1      3.6   
981    39    1   0       118   219    0        1      140      0      1.2   
973    51    1   2       125   245    1        0      166      0      2.4   

     slope  ca  thal  target  
960      1   0     2       1  
48       1   1     2       1  
886      1   1     3       0  
981      1   0     3       0  
973      1   0     2       1  
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 960 to 872
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       200 non-null    int64  
 1   sex       200 non-null    int64  
 2   cp        200 non-null    int64  
 3   trestbps  200 non-null    int64  
 4   chol      200 non-null    int64  
 5   fbs       200 non-null    int64  
 6   restecg   200 non-null    int64  
 7   thalach   200 non-null    int64  
 8   exang     200 non-null    int64  
 9   oldpeak   200 non-null    float64
 10  slope     200 non-null    int64  
 11  ca        200 non-null    int64  
 12  thal      200 non-null    int64  
 13  target    200 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 23.4 KB
None
```

2. Stratified Sampling

- **Definition:** In stratified sampling, the population is divided into subgroups (strata) based on certain characteristics, and then a random sample is selected from each stratum.
- **Use Cases:** When the population is heterogeneous and it is important to ensure that all subgroups are represented in the sample.
- **Advantages:** Guarantees the representation of different subgroups and allows for comparisons between subgroups.
- **Disadvantages:** Requires prior knowledge of the population's characteristics and can be more complex to implement.
- **Example (Heart Disease Dataset):** Dividing the dataset into strata based on age groups (e.g., 20-30, 30-40, etc.) and selecting a random sample from each age group to ensure that all age groups are represented.

In [5]:
```
# Assuming 'age' is a column in your DataFrame (df)
# Create age groups (strata)
bins = [0, 20, 30, 40, 50, 60, 70, 80]  # Adjust bins as needed
labels = ['0-19', '20-29', '30-39', '40-49', '50-59', '60-69', '70+']
df['age_group'] = pd.cut(df['age'], bins=bins, labels=labels, include_lowest=True)

# Stratified sampling with 10 samples per age group
stratified_sample = df.groupby('age_group', group_keys=False).apply(lambda x: x.sample(min(len(x), 10)))

# Now, 'stratified_sample' contains your stratified sample
stratified_sample
```

Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	age_group
118	29	1	1	130	204	0	0	202	0	0.0	2	0	2	1	20-29
668	29	1	1	130	204	0	0	202	0	0.0	2	0	2	1	20-29
64	29	1	1	130	204	0	0	202	0	0.0	2	0	2	1	20-29
60	29	1	1	130	204	0	0	202	0	0.0	2	0	2	1	20-29
979	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1	30-39
340	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1	30-39
86	38	1	0	110	167	0	0	114	1	2.0	1	0	3	0	30-39
701	35	1	0	120	198	0	1	130	1	1.6	1	0	3	0	30-39
52	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1	30-39
314	40	1	3	140	199	0	1	178	1	1.4	2	0	3	1	30-39
156	40	1	3	140	199	0	1	178	1	1.4	2	0	3	1	30-39
272	39	0	2	138	220	0	1	152	0	0.0	1	0	2	1	30-39
242	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1	30-39
254	35	1	0	120	198	0	1	130	1	1.6	1	0	3	0	30-39
67	42	1	0	136	315	0	1	125	1	1.8	1	0	1	0	40-49
400	49	1	2	120	188	0	1	139	0	2.0	1	3	3	0	40-49
774	48	0	2	130	275	0	1	139	0	0.2	2	0	2	1	40-49
8	46	1	0	120	249	0	0	144	0	0.8	2	0	3	0	40-49
1019	47	1	0	112	204	0	1	143	0	0.1	2	0	2	1	40-49
857	43	1	0	115	303	0	1	181	0	1.2	1	0	2	1	40-49
1018	41	1	0	110	172	0	0	158	0	0.0	2	0	3	0	40-49
855	46	1	1	101	197	1	1	156	0	0.0	2	0	3	1	40-49
214	45	1	1	128	308	0	0	170	0	0.0	2	0	2	1	40-49
771	45	0	0	138	236	0	0	152	1	0.2	1	0	2	1	40-49
13	51	1	0	140	298	0	1	122	1	4.2	1	3	3	0	50-59
28	56	1	2	130	256	1	0	142	1	0.6	1	1	1	0	50-59
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1	50-59
843	59	1	3	160	273	0	0	125	0	0.0	2	0	2	0	50-59
356	59	1	0	164	176	1	0	90	0	1.0	1	2	1	0	50-59
97	53	1	0	123	282	0	1	95	1	2.0	1	2	3	0	50-59
268	58	1	2	132	224	0	0	173	0	3.2	2	2	3	0	50-59
804	58	0	0	130	197	0	1	131	0	0.6	1	0	2	1	50-59
383	58	1	0	150	270	0	0	111	1	0.8	2	0	3	0	50-59
264	58	0	0	170	225	1	0	146	1	2.8	1	2	1	0	50-59
496	68	1	2	180	274	1	0	150	1	1.6	1	0	3	0	60-69
116	63	1	0	130	254	0	0	147	0	1.4	1	1	3	0	60-69
47	66	0	0	178	228	1	1	165	1	1.0	1	2	3	0	60-69
480	61	1	2	150	243	1	1	137	1	1.0	1	0	2	1	60-69
266	64	0	2	140	313	0	1	133	0	0.2	2	0	3	1	60-69
229	66	0	0	178	228	1	1	165	1	1.0	1	2	3	0	60-69
861	64	1	2	140	335	0	1	158	0	0.0	2	0	2	0	60-69
884	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0	60-69
378	67	1	0	120	237	0	1	71	0	1.0	1	0	2	0	60-69
542	62	0	0	160	164	0	0	157	0	1.2	1	0	2	1	60-69
965	76	0	2	140	197	0	2	116	0	1.1	1	0	2	1	70+
989	71	0	1	160	302	0	1	162	0	0.4	2	2	2	1	70+
724	74	0	1	120	269	0	0	121	1	0.2	2	1	2	1	70+
648	71	0	0	112	149	0	1	125	0	1.6	1	0	2	1	70+
99	70	0	2	140	197	0	2	116	0	1.1	1	0	2	1	70+
535	76	0	2	140	197	0	2	116	0	1.1	1	0	2	1	70+
287	71	0	1	160	302	0	1	162	0	0.4	2	2	2	1	70+
269	71	0	2	110	265	1	0	130	0	0.0	2	1	2	1	70+
285	71	0	2	110	265	1	0	130	0	0.0	2	1	2	1	70+
401	71	0	0	112	149	0	1	125	0	1.6	1	0	2	1	70+

Interpreting the Code:

Stratified sampling with 10 samples per age group:

stratified_sample = df.groupby('age_group', group_keys=False).apply(lambda x: x.sample(min(len(x), 10)))

df.groupby('age_group', group_keys=False): This part groups your DataFrame (df) based on the 'age_group' column we just created. It essentially creates separate subsets of data for each age group. group_keys=False prevents the group keys from becoming an index in the result.

.apply(lambda x: x.sample(min(len(x), 10))): This line applies a function to each age group subset.

The lambda function is a way to define a small, anonymous function within the code. In this case, it takes one argument (x), which represents each age group subset.

x.sample(min(len(x), 10)): This part randomly selects a sample from the current age group subset (x). It ensures you get a maximum of 10 samples or fewer if the age group has less than 10 individuals.

stratified_sample: The result of the stratified sampling, your representative sample, is stored in a new DataFrame called stratified_sample.

3. Cluster Sampling

- **Definition:** In cluster sampling, the population is divided into clusters, and then a random sample of clusters is selected. All individuals within the selected clusters are then included in the sample.
- **Use Cases:** When the population is geographically dispersed or when it is difficult to identify individual members of the population.
- **Advantages:** Cost-effective and easier to implement than other sampling methods.
- **Disadvantages:** May not be as representative of the population as other methods, as the sample may be influenced by the characteristics of the selected clusters.
- **Example (Heart Disease Dataset):** Dividing the dataset into clusters based on hospitals or clinics where the patients were treated. Then, randomly selecting a few hospitals/clinics and including all patients from those selected hospitals/clinics in the sample.

4. Systematic Sampling

- **Definition:** In systematic sampling, individuals are selected from the population at regular intervals (e.g., every 10th person).
- **Use Cases:** When the population is ordered in some way and it is important to ensure that all parts of the population are represented.
- **Advantages:** Easy to implement and can be more efficient than other methods.
- **Disadvantages:** May not be representative of the population if the population is not randomly ordered.
- **Example (Heart Disease Dataset):** Selecting every 5th patient from the dataset to create a sample. This assumes that the dataset is not ordered in a way that would bias the sample.

In [6]:
```
# Select every 5th patient
systematic_sample = df.iloc[::5]

# Print the first few rows of the systematic sample
print(systematic_sample.head())

print(systematic_sample.info())

# You can now perform further analysis or modeling on this sample
```

```
    age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0    52    1   0       125   212    0        1      168      0      1.0   
5    58    0   0       100   248    0        0      122      0      1.0   
10   71    0   0       112   149    0        1      125      0      1.6   
15   34    0   1       118   210    0        1      192      0      0.7   
20   60    1   2       140   185    0        0      155      0      3.0   

     slope  ca  thal  target age_group  
0        2   2     3       0     50-59  
5        1   0     2       1     50-59  
10       1   0     2       1       70+  
15       2   0     2       1     30-39  
20       1   0     2       0     50-59  
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 1020
Data columns (total 15 columns):
 #   Column     Non-Null Count  Dtype   
---  ------     --------------  -----   
 0   age        205 non-null    int64   
 1   sex        205 non-null    int64   
 2   cp         205 non-null    int64   
 3   trestbps   205 non-null    int64   
 4   chol       205 non-null    int64   
 5   fbs        205 non-null    int64   
 6   restecg    205 non-null    int64   
 7   thalach    205 non-null    int64   
 8   exang      205 non-null    int64   
 9   oldpeak    205 non-null    float64 
 10  slope      205 non-null    int64   
 11  ca         205 non-null    int64   
 12  thal       205 non-null    int64   
 13  target     205 non-null    int64   
 14  age_group  205 non-null    category
dtypes: category(1), float64(1), int64(13)
memory usage: 23.1 KB
None
```

In [ ]:
```

```