Sampling Battery Range using Exponential Distribution

Part A: Sampling Battery Range

Q1: Generate a Population of 10,000 EV battery ranges.

We use an exponential distribution with mean approximately 280 km.
For an exponential distribution:

$$\text{Mean} = \frac{1}{\lambda} \Rightarrow \lambda = \frac{1}{280}$$

Python Code:

```
import numpy as np
import matplotlib.pyplot as plt

# Set random seed for reproducibility
np.random.seed(42)

# Parameters
mean_range = 280
lambda_param = 1 / mean_range
population_size = 10000

# Generate the population
population = np.random.exponential(scale=mean_range, size
    =population_size)
```

Q2: Draw 1000 Random Samples of size 30 each.

We simulate 1000 samples of size 30 from the exponential population and compute the mean of each sample.

Python Code:

```
sample_size = 30
num_samples = 1000

# Draw samples and compute means
sample_means = [
    np.random.choice(population, size=sample_size,
        replace=False).mean()
    for _ in range(num_samples)
]
```

Q3: Plot the Histogram of Sample Means.What shape we see?

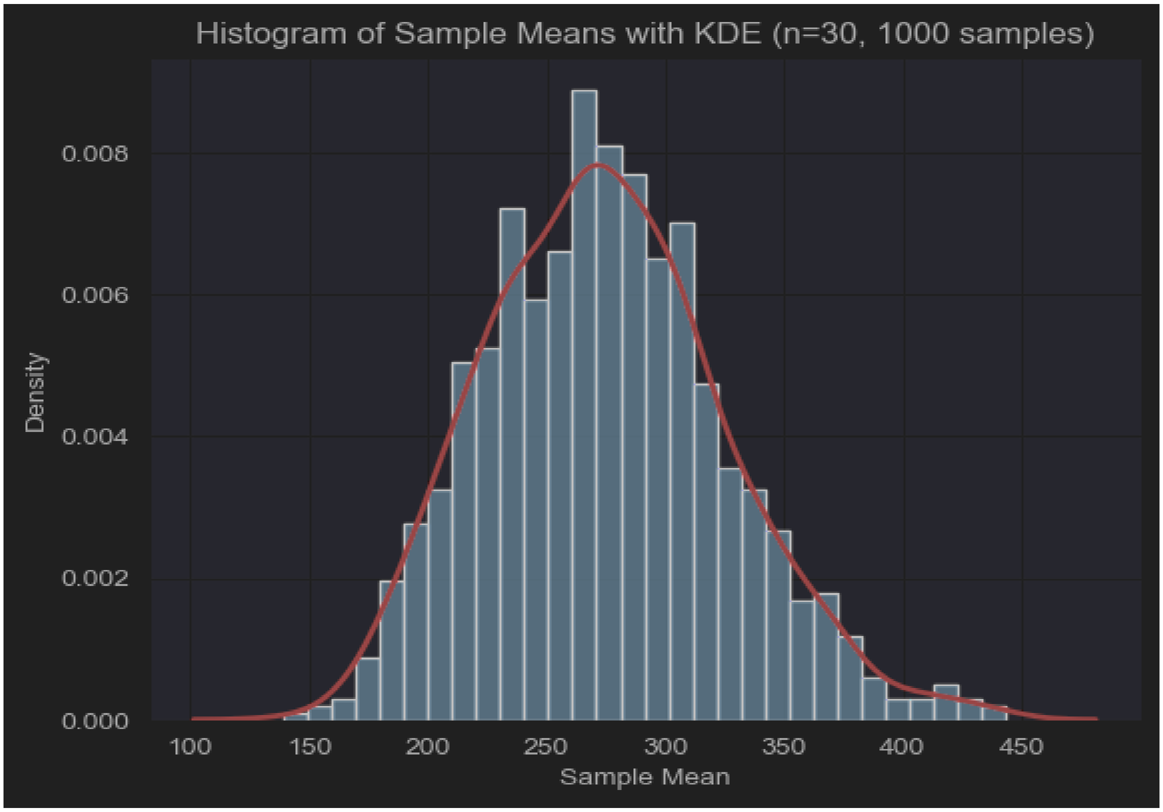
We use a histogram to observe the shape of the sampling distribution.

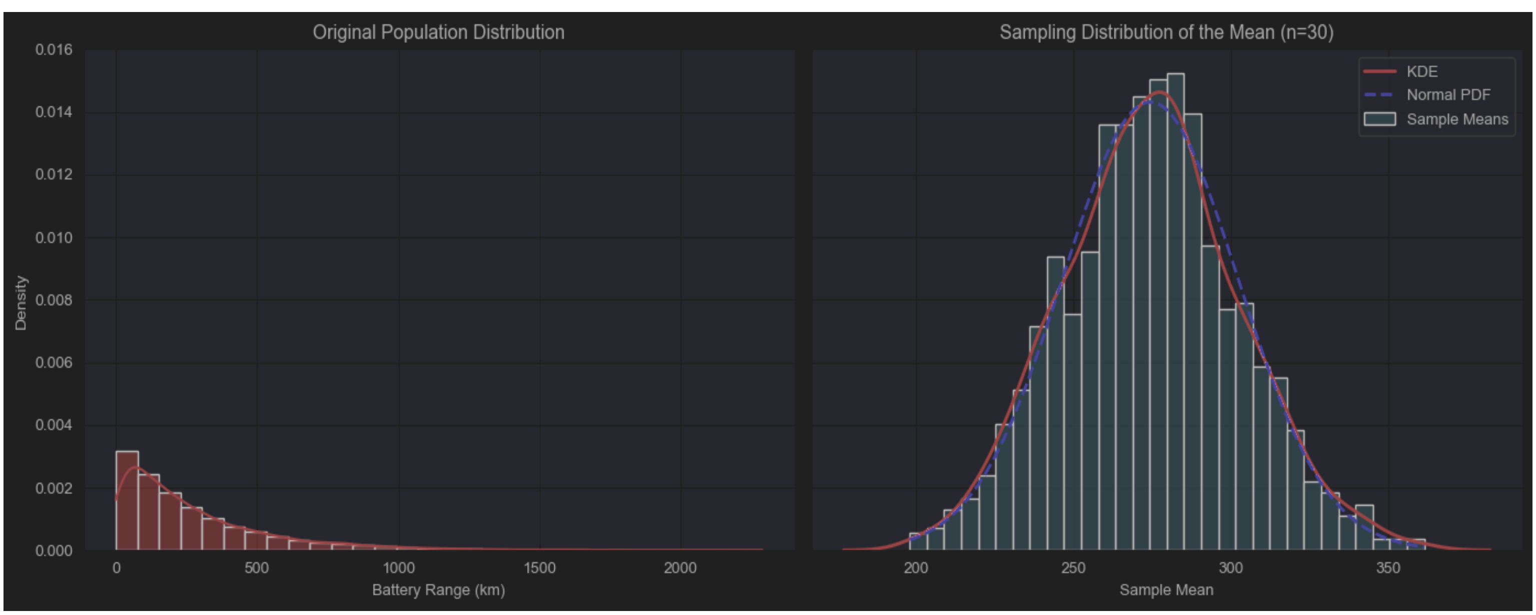
Python Code:

```
import seaborn as sns

# Plot histogram and KDE
sns.histplot(sample_means, bins=30, edgecolor='black',
    stat='density')
sns.kdeplot(sample_means, color='red', linewidth=2)

plt.title('Histogram of Sample Means with KDE (n=30, 1000
     samples)')
plt.xlabel('Sample Mean')
plt.ylabel('Density')
plt.grid(True)
plt.show()
```

Observations:

- **Bell-Shaped Curve:** The KDE shows a symmetric, bell-shaped distribution even though the population was skewed.
- **Centered Around ∼280 km:** The sample means are centered around the population mean, indicating an unbiased estimator.
- **Reduced Variability:** Sample means are more tightly clustered than the original population.


Histogram of Sample Means with KDE (n=30, 1000 samples)
Density
0.008
0.006
0.004
0.002
0.000
100
150
200
250
300
350
400
450
Sample Mean


Original Population Distribution
Sampling Distribution of the Mean (n=30)
0.016
0.014
0.012
0.010
0.008
0.006
0.004
0.002
0.000
Density
0
500
1000
1500
2000
Battery Range (km)
KDE
Normal PDF
Sample Means
200
250
300
350
Sample Mean

Population vs Sample Mean Distribution

Left Plot: Original Population Distribution

- Simulated from an exponential distribution (heavily right-skewed).
- Most EVs have lower ranges (0–400 km).
- Some EVs have ranges over 2000 km.

Right Plot: Sampling Distribution of Means (n = 30)

- Distribution of 1000 sample means is bell-shaped.
- Red KDE line: Empirical sampling distribution.
- Blue dashed line (optional): Normal distribution with same mean/std.
- Visual confirmation of the Central Limit Theorem (CLT).

Q4: How This Relates to the Central Limit Theorem (CLT)

The Central Limit Theorem (CLT) states:
If you take sufficiently large, random samples from **any** population (regardless of the original distribution), the distribution of the **sample means** will approximate a **normal distribution** as sample size increases.

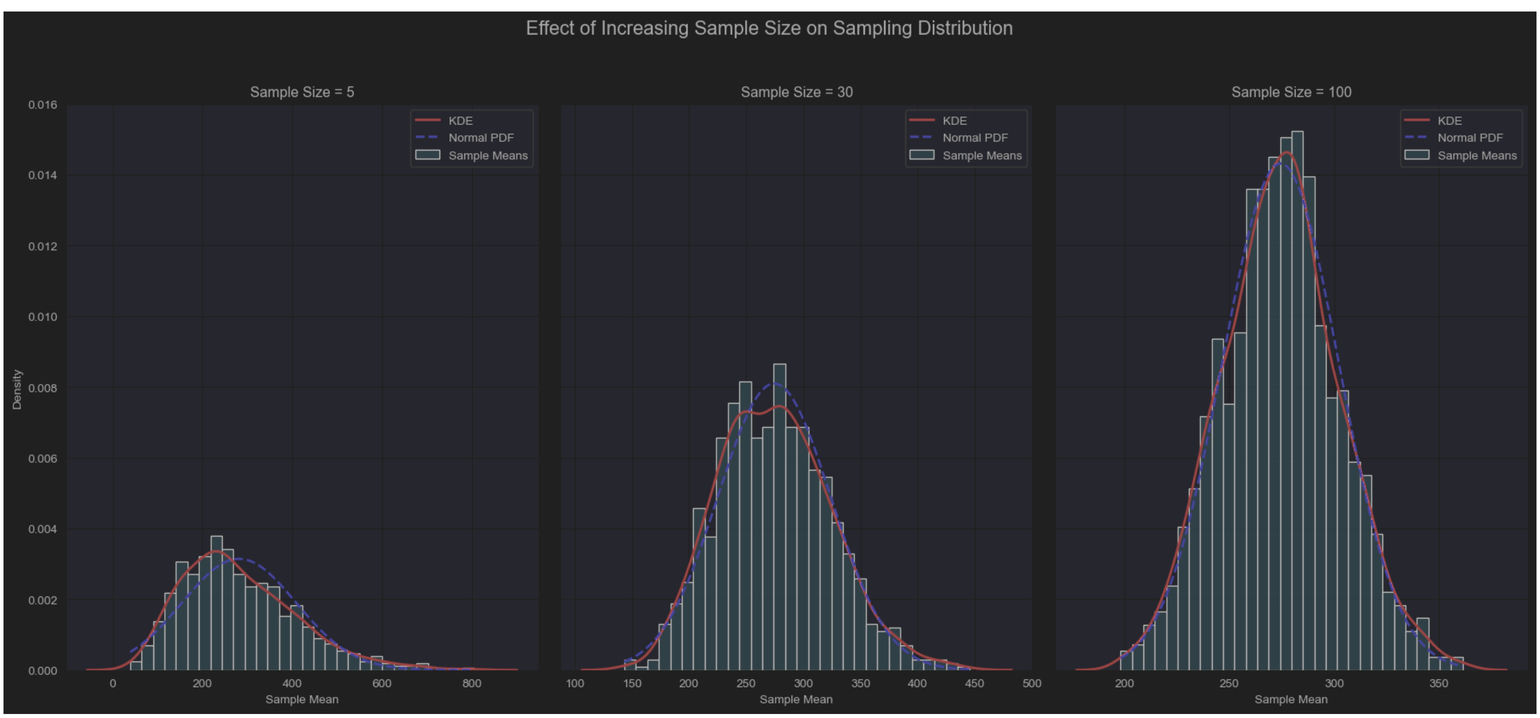
Step-by-step CLT demonstration:

Step	What We Did	How It Relates to CLT
1	Generated a population from an exponential distribution (very skewed)	Proves CLT applies to non-normal populations
2	Drew 1000 random samples of size 30	Demonstrates repeated sampling
3	Plotted histogram of sample means	Bell-shaped curve confirms the sampling distribution is approximately normal
4	Overlaid KDE and optional normal curve	Visual confirmation of CLT behavior

Part B: Effect of Sample Size on Sampling Distribution

Q5: Simulate the Effect of Increasing Sample Size

We repeat the previous sampling process, but vary the sample size: $n = 5$, $n = 30$, and $n = 100$. We observe how the sampling distribution of the mean changes with increasing sample size.



Python Code for Simulation and Plotting

```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import norm

# Simulate original exponential population
np.random.seed(42)
population = np.random.exponential(scale=280, size=10000)

# Sample sizes to test
sample_sizes = [5, 30, 100]
fig, axes = plt.subplots(1, 3, figsize=(18, 8), sharey=
    True)

for i, n in enumerate(sample_sizes):
    # Draw 1000 samples of size n
    sample_means = [np.mean(np.random.choice(population,
        size=n, replace=True)) for _ in range(1000)]

    # Plot histogram
    sns.histplot(sample_means, bins=30, stat='density',
        edgecolor='black', ax=axes[i], color='lightblue',
         label='Sample Means')

    # KDE
    sns.kdeplot(sample_means, color='red', linewidth=2,
        ax=axes[i], label='KDE')

    # Normal curve
    mean = np.mean(sample_means)
    std = np.std(sample_means)
    x = np.linspace(min(sample_means), max(sample_means),
         1000)
    y = norm.pdf(x, loc=mean, scale=std)
    axes[i].plot(x, y, color='blue', linestyle='--',
        linewidth=2, label='Normal PDF')

    # Formatting
    axes[i].set_title(f'Sample Size = {n}')
    axes[i].set_xlabel('Sample Mean')
    if i == 0:
        axes[i].set_ylabel('Density')
    axes[i].legend()
    axes[i].grid(True)

plt.suptitle('Effect of Increasing Sample Size on
    Sampling Distribution', fontsize=16)
plt.tight_layout(rect=[0, 0, 1, 0.95])
plt.show()
```

Q6: What Happens as Sample Size Increases?

Summary Table:

Sample Size	Shape of Sampling Distribution
$n = 5$	Wide spread, more variability, slight right-skew.
$n = 30$	Narrower spread, smoother, bell-shaped.
$n = 100$	Very tight distribution, very close to normal.

Interpretation: As the sample size increases:

- The sampling distribution becomes **narrower**.
- It becomes more **symmetric and normal-shaped**.
- The sample mean becomes a more **precise estimator** of the population mean.

This supports the **Central Limit Theorem** which predicts that the distribution of sample means becomes more normal as sample size grows — even if the original population is skewed.

Q7: How Does the Spread (Standard Deviation) Change?

The spread of the sampling distribution of the sample mean is known as the **Standard Error of the Mean (SEM)**.

$$\text{SEM} = \frac{\sigma}{\sqrt{n}}$$

Where:

- σ = standard deviation of the population.
- n = sample size.

Conclusion:

- As n increases, SEM decreases.
- This results in a **tighter distribution** of sample means.
- Larger samples provide **more stable and reliable** estimates of the population mean.

Q8: Risks of Small Samples in Engineering Decisions

Why Sample Size Matters in Engineering

Context: You want to estimate the average EV charging time using only **10 samples**. This raises important concerns about **statistical reliability**, especially in critical fields like **engineering**, where poor estimates can lead to flawed decisions.

Risks of Using Small Samples in Engineering Decisions

1. High Variability / Unreliable Estimates

- Small samples tend to have **higher variance**, making the sample mean an unreliable estimator.
- You may significantly **underestimate or overestimate** the true average charging time.

2. Lack of Representativeness

- Small samples are more likely to **miss the population's diversity** — e.g., slow vs fast chargers, different battery capacities.
- This may lead to **biased or skewed** results.

3. Greater Influence of Outliers

- A single **extreme value** can greatly affect the mean in small samples.
- *Example:* One ultra-fast charger in the sample could **artificially lower** the estimated charging time.

4. Reduced Confidence

- Small samples produce **wider confidence intervals**, reducing the precision of your estimate.
- Difficult to **justify design or policy decisions** with uncertain results.

5. Misleading Normality Assumption

- The **Central Limit Theorem (CLT)** doesn't fully apply at small n, so assuming normality in inference (e.g., confidence intervals, hypothesis tests) is **risky**.

Short Summary (for Reports or Slides)

Using small samples (e.g., 10) to estimate engineering metrics like EV charging time can lead to ***unreliable, biased, and highly variable estimates****. These are* ***sensitive to outliers*** *and offer* ***low confidence*** *for making informed decisions. Larger, representative samples are essential for drawing* ***robust and valid engineering conclusions****.*

Q9: Why the i.i.d. Assumption Matters in Engineering

What Does i.i.d. Mean?

Independent and Identically Distributed (i.i.d.) samples satisfy two key conditions:

- **Independent:** Each observation is collected without influence from others (no autocorrelation).
- **Identically Distributed:** All observations come from the same underlying probability distribution (same mean, variance, etc.).

Why Is This Important in Engineering?

1. Valid Statistical Inference

- Most statistical techniques — including confidence intervals, hypothesis tests, and regression — **assume i.i.d. data**.
- Violations can lead to **biased estimates** or **invalid conclusions**.

2. Sensor Readings Example

- If sensor data is **not independent** (e.g., time-correlated due to drift), you may detect **illusory patterns**.
- If it's **not identically distributed** (e.g., due to environmental differences), you're mixing *apples and oranges*.

3. Accurate Estimation

- Estimating population parameters (e.g., mean or variance) requires data that behaves **consistently (identical)** and **unbiasedly (independent)**.
- *Example:* Calibrating EV battery sensors with non-i.i.d. data could yield **unsafe or faulty charge estimates**.

4. Model Validity

- Engineering systems (e.g., ML models, control systems, anomaly detection) rely on i.i.d. data to ensure **stability and generalization**.

Summary for Report or Slide

The i.i.d. assumption is critical in engineering as it ensures data is ***consistent, reliable, and statistically valid****. Violating this assumption can result in* ***flawed inferences, unstable models, and unsafe decisions****, particularly in high-stakes systems like* ***sensor networks, process control, or vehicle diagnostics****.*

Q10: Explaining the Central Limit Theorem (CLT) to a Non-Technical EV Engineer

Simple Explanation of the Central Limit Theorem (CLT)

Imagine you're checking EV battery range, charging times, or sensor temperatures... and the readings vary a lot.

The **Central Limit Theorem** says that **if you take enough small, random samples and average them**, those averages will start to follow a **predictable, bell-shaped (normal) pattern** *even if the original data is messy, skewed, or noisy.*

Real-World Analogy for an EV Engineer

Think of each data point like a car's charging time on a different day. One day might be fast, another slow ... it's unpredictable.

But if you average 30 cars' times every day, those daily averages become **much more stable** and **easier to work with** — that's the CLT in action.

Why It Matters for Real-Time EV Monitoring

Use Case	CLT Insight
Battery Health Monitoring	Helps average out noisy readings to give a reliable estimate of battery performance.
Charging Network Analysis	Allows you to estimate average charging times confidently, even with variability across locations.
Temperature Sensor Stability	Enables filtering out sensor "blips" — averaged readings become smooth and normally distributed.
Predictive Maintenance	Reliable averages from small sensor samples can signal when a component is drifting out of spec.

Summary for a Slide or Report

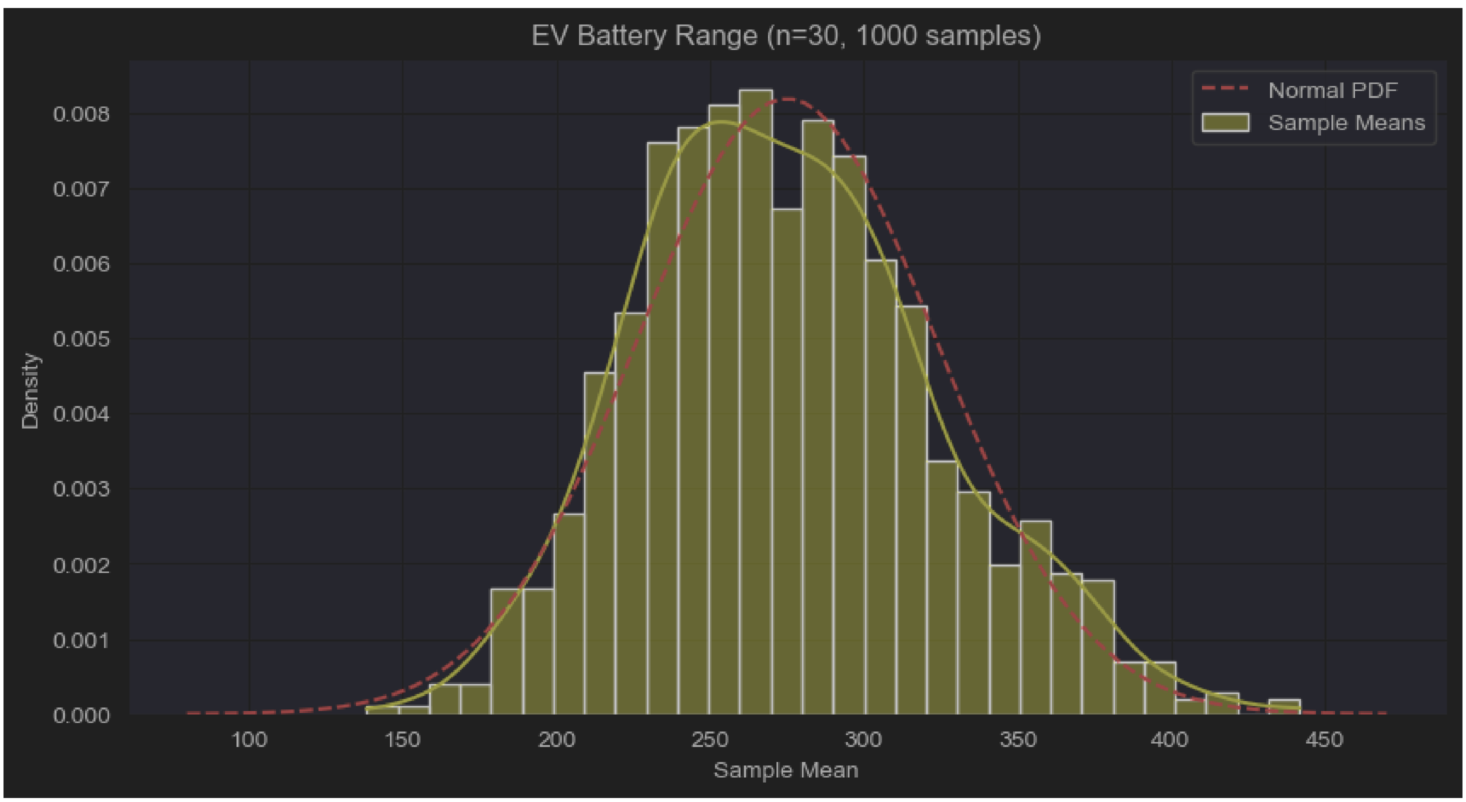
*The Central Limit Theorem (CLT) helps engineers turn noisy, real-world data into **stable, predictable averages**. Even if individual EV readings are messy, the average of a few can be trusted — enabling **better real-time monitoring, diagnostics, and decision-making**.*

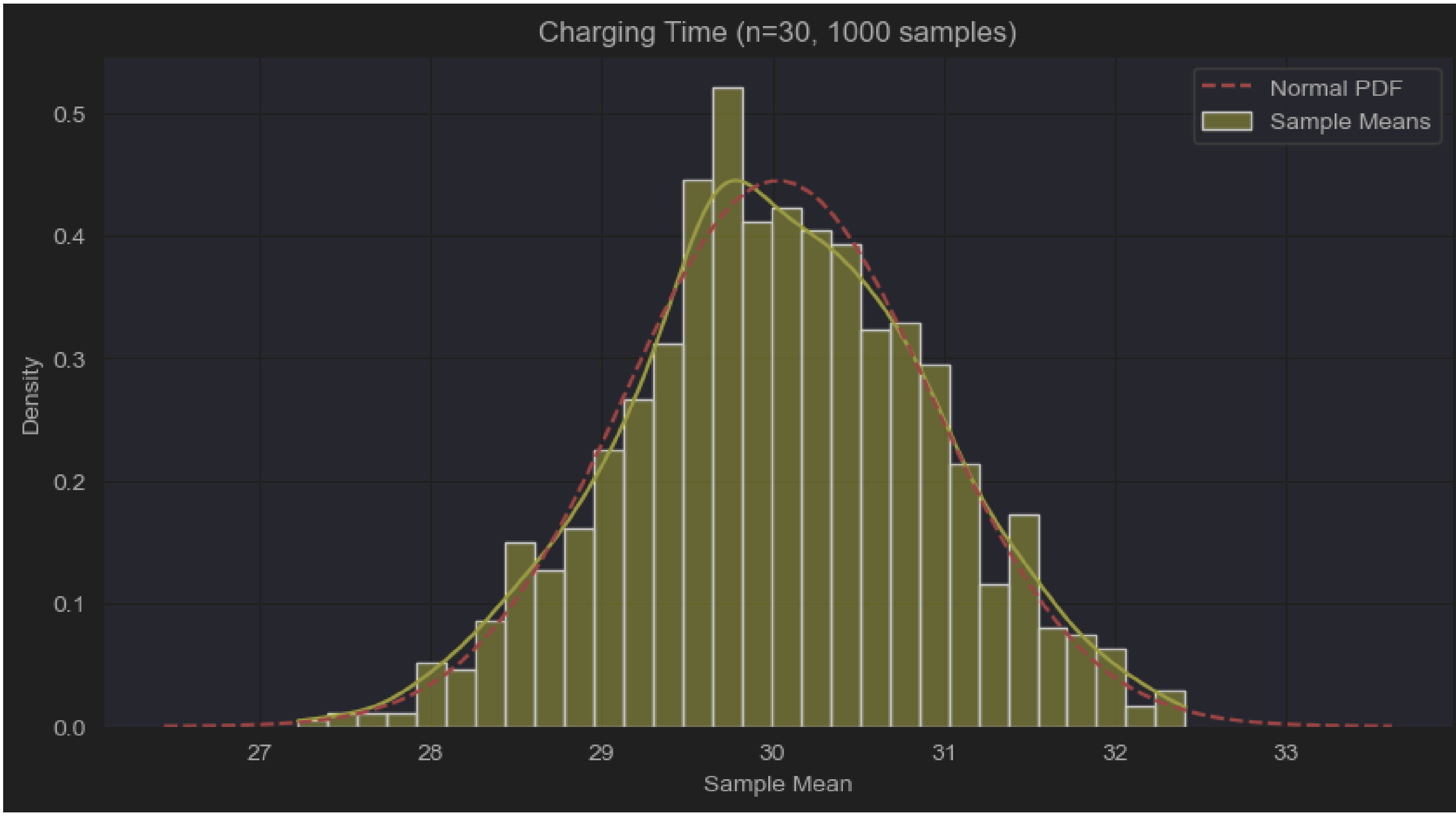
Challenge Task: Sampling Distribution Function in Python

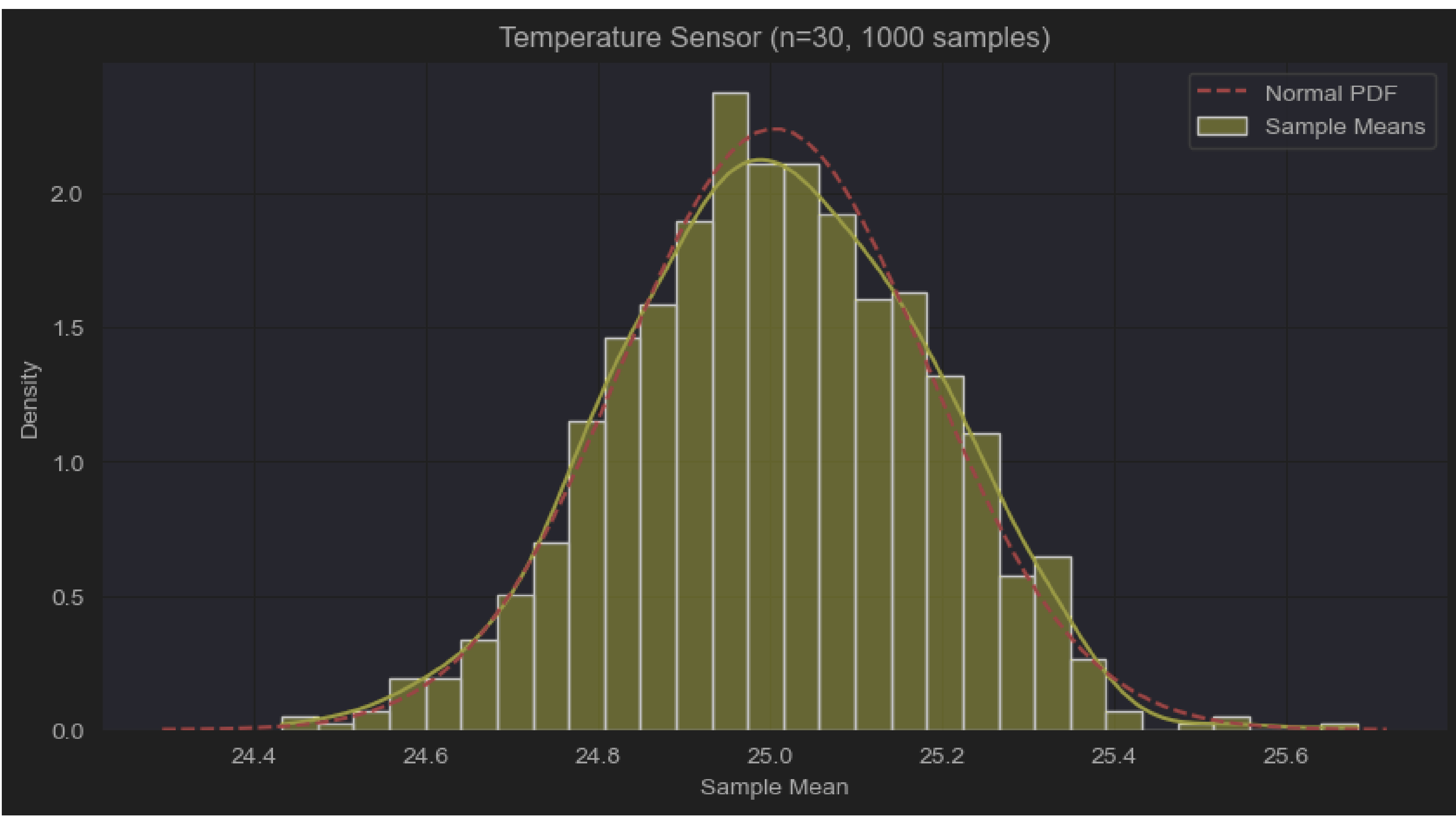
Objective

Create a Python function that takes a population, sample size, and number of samples, then returns the sampling distribution and a histogram plot. Use this to compare:

- EV battery range
- Charging time
- Temperature sensor readings




Charging Time (n=30, 1000 samples)
Normal PDF
Sample Means
Density
0.0
0.1
0.2
0.3
0.4
0.5
27
28
29
30
31
32
33
Sample Mean


Temperature Sensor (n=30, 1000 samples)
Normal PDF
Sample Means
Density
0.0
0.5
1.0
1.5
2.0
24.4
24.6
24.8
25.0
25.2
25.4
25.6
Sample Mean

Python Function

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

def sampling_distribution(population, sample_size=30, num_samples=1000, title='Sampling Distribution'):
    """
    Simulate a sampling distribution from a given population.

    Parameters:
    - population (array-like): The full population data.
    - sample_size (int): Number of observations per sample.
    - num_samples (int): Number of samples to draw.
    - title (str): Plot title.

    Returns:
    - sample_means (np.ndarray): The means of the sampled data.
    """
    sample_means = []

    for _ in range(num_samples):
        sample = np.random.choice(population, size=sample_size, replace=True)
        sample_means.append(np.mean(sample))

    sample_means = np.array(sample_means)

    # Plot
    plt.figure(figsize=(10, 5))
    sns.histplot(sample_means, bins=30, kde=True, color='yellow', edgecolor='black', stat='density',
         label='Sample Means')

    # Overlay normal curve
    mean = np.mean(sample_means)
    std = np.std(sample_means)
    x = np.linspace(mean - 4*std, mean + 4*std, 100)
    plt.plot(x,
             (1 / (std * np.sqrt(2 * np.pi))) * np.exp(-0.5 * ((x - mean)/std)**2),
             color='red', linestyle='--', label='Normal PDF')

    plt.title(f'{title} (n={sample_size}, {num_samples} samples)')
    plt.xlabel('Sample Mean')
    plt.ylabel('Density')
    plt.legend()
    plt.grid(True)
    plt.show()

    return sample_means
```

Data Simulation for Use Cases

```
# Simulated data for demonstration
np.random.seed(42)

# 1. EV range: exponential (skewed)
ev_range = np.random.exponential(scale=280, size=10000)

# 2. Charging time: normal-like (e.g., 30 min  5 min)
charging_time = np.random.normal(loc=30, scale=5, size=10000)

# 3. Temperature sensor: tight Gaussian (e.g., 25C  1C)
temperature = np.random.normal(loc=25, scale=1, size=10000)
```

Function Application for Comparisons

```
# Run comparisons
sampling_distribution(ev_range, sample_size=30, title='EV␣Battery␣Range')
sampling_distribution(charging_time, sample_size=30, title='Charging␣Time')
sampling_distribution(temperature, sample_size=30, title='Temperature␣Sensor')
```

Summary

This function demonstrates how sampling distributions change depending on the nature of the population. It reinforces the Central Limit Theorem by showing that even non-normal populations (like EV battery range) yield approximately normal sampling distributions when sample size is sufficiently large.