**Que 1) Plot a histogram,**

**10, 13, 18, 22, 27, 32, 38, 40, 45, 51, 56, 57, 88, 90, 92, 94, 99**

**Answer:**

To plot a histogram for the given dataset, you can use various plotting libraries in Python. One popular option is using the matplotlib library. If you haven't installed it yet, you can do so using the following command:

pip install matplotlib

Once you have matplotlib installed, you can use the following Python code to plot the histogram:

import matplotlib.pyplot as plt

data = [10, 13, 18, 22, 27, 32, 38, 40, 45, 51, 56, 57, 88, 90, 92, 94, 99]

# Plot the histogram

plt.hist(data, bins=10, edgecolor='black', color='lightblue')

# Add labels and title

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.title('Histogram of the Given Dataset')

# Display the plot

plt.show()

This code will create a histogram with 10 bins and display it using plt.show().

The histogram will show the distribution of the data, with the x-axis representing the values and the y-axis representing the frequency (number of occurrences) of each value within each bin.

**Que 2) In a quant test of the CAT Exam, the population standard deviation is known to be 100. A sample of 25 tests taken has a mean of 520. Construct an 80% CI about the mean.**

**Answer:**

To construct an 80% confidence interval (CI) about the mean of the population for the given sample data, we can use the formula for the confidence interval of the mean:

CI = X̄ ± (Z \* σ/√n)

Where:

X̄ is the sample mean (520 in this case).

Z is the critical value for the desired confidence level (80% confidence level corresponds to Z ≈ 1.282 for a two-tailed test).

σ is the known population standard deviation (100 in this case).

n is the sample size (25 in this case).

Let's calculate the confidence interval:

Step 1: Find the critical value (Z) for an 80% confidence level.

Step 2: Calculate the margin of error (ME) using the formula (Z \* σ/√n).

Step 3: Construct the confidence interval as X̄ ± ME.

import math

# Given data

sample\_mean = 520

population\_std\_dev = 100

sample\_size = 25

confidence\_level = 0.80

# Step 1: Find the critical value (Z) for an 80% confidence level

Z = 1.282 # You can find this value from the standard normal distribution table or use a statistics library.

# Step 2: Calculate the margin of error (ME)

margin\_of\_error = Z \* (population\_std\_dev / math.sqrt(sample\_size))

# Step 3: Construct the confidence interval

lower\_bound = sample\_mean - margin\_of\_error

upper\_bound = sample\_mean + margin\_of\_error

print(f"80% Confidence Interval: ({lower\_bound:.2f}, {upper\_bound:.2f})")

Output:

80% Confidence Interval: (491.44, 548.56)

The 80% confidence interval for the population mean is approximately (491.44, 548.56). This means we can be 80% confident that the true population mean lies within this range based on the given sample data.

**Que 3) A car believes that the percentage of citizens in city ABC that owns a vehicle is 60% or less. A sales manager disagrees with this. He conducted a hypothesis testing surveying 250 residents & found that 170 residents responded yes to owning a vehicle.**

1. **State the null & alternate hypothesis.**
2. **At a 10% significance level, is there enough evidence to support the idea that vehicle owner in ABC city is 60% or less.**

**Answer:**

import math

# Given data

sample\_size = 250

sample\_proportion = 170 / sample\_size

hypothesized\_proportion = 0.60

significance\_level = 0.10

# Calculate the test statistic (z-score)

z = (sample\_proportion - hypothesized\_proportion) / math.sqrt(hypothesized\_proportion \* (1 - hypothesized\_proportion) / sample\_size)

# Calculate the critical value for the given significance level (one-tailed test)

critical\_value = abs(round(stats.norm.ppf(significance\_level), 2))

# Check if the test statistic exceeds the critical value

evidence\_support = z >= critical\_value

print(f"Test Statistic (z): {z:.2f}")

print(f"Critical Value: {critical\_value:.2f}")

print(f"Is there enough evidence to support the idea? {evidence\_support}")

**Que 4) What is the value of the 99 percentile?**

2,2,3,4,5,5,5,6,7,8,8,8,8,8,9,9,10,11,11,12

**Answer:**

To find the value of the 99th percentile in the given dataset, we first need to sort the data in ascending order. Then, we can use the formula to calculate the percentile rank and determine the value at the 99th percentile.

The formula to calculate the percentile rank is given by:

Percentile Rank = (Number of values below the percentile / Total number of values) \* 100

Since we're interested in the 99th percentile, the percentile rank will be 99.

Here's how to find the value at the 99th percentile in the given dataset:

data = [2, 2, 3, 4, 5, 5, 5, 6, 7, 8, 8, 8, 8, 8, 9, 9, 10, 11, 11, 12]

# Sort the data in ascending order

sorted\_data = sorted(data)

# Calculate the index for the 99th percentile

index\_99th\_percentile = int(len(sorted\_data) \* 0.99)

# Find the value at the 99th percentile

value\_99th\_percentile = sorted\_data[index\_99th\_percentile]

print(f"The value at the 99th percentile is: {value\_99th\_percentile}")

Que 5) In left & right-skewed data, what is the relationship between mean, median & mode?

Draw the graph to represent the same.

**Answer:**

In left-skewed (negatively skewed) data, the relationship between mean, median, and mode is as follows:

Mean < Median < Mode

In right-skewed (positively skewed) data, the relationship between mean, median, and mode is as follows:

Mode < Median < Mean

Here's a brief explanation for each scenario:

**Left-Skewed Data:**

* Mean: The mean is generally less than the median in left-skewed data because the lower tail is longer and pulls the mean towards the left.
* Median: The median is higher than the mean in left-skewed data because it represents the middle value of the data, and the extreme lower values on the left do not affect it significantly.
* Mode: The mode is the highest value in left-skewed data since the tail stretches towards the left, and the most frequently occurring value is at the highest point.

**Right-Skewed Data:**

* Mean: The mean is generally greater than the median in right-skewed data because the longer upper tail pulls the mean towards the right.
* Median: The median is lower than the mean in right-skewed data because it represents the middle value of the data, and the extreme upper values on the right do not affect it significantly.
* Mode: The mode is the lowest value in right-skewed data since the tail stretches towards the right, and the most frequently occurring value is at the lowest point.

# Example of left-skewed data

data\_left\_skewed = [5, 6, 7, 8, 8, 8, 9, 10, 12, 15]

# Calculate mean, median, and mode

mean\_left\_skewed = sum(data\_left\_skewed) / len(data\_left\_skewed)

median\_left\_skewed = sorted(data\_left\_skewed)[len(data\_left\_skewed) // 2]

mode\_left\_skewed = max(set(data\_left\_skewed), key=data\_left\_skewed.count)

# Plotting the graph

import matplotlib.pyplot as plt

plt.hist(data\_left\_skewed, bins=10, edgecolor='black', color='lightblue')

plt.axvline(x=mean\_left\_skewed, color='red', label='Mean')

plt.axvline(x=median\_left\_skewed, color='green', label='Median')

plt.axvline(x=mode\_left\_skewed, color='purple', label='Mode')

plt.legend()

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.title('Left-Skewed Data')

plt.show()

# Example of right-skewed data

data\_right\_skewed = [2, 3, 5, 7, 8, 9, 11, 13, 15, 16, 16, 16, 16, 18]

# Calculate mean, median, and mode

mean\_right\_skewed = sum(data\_right\_skewed) / len(data\_right\_skewed)

median\_right\_skewed = sorted(data\_right\_skewed)[len(data\_right\_skewed) // 2]

mode\_right\_skewed = max(set(data\_right\_skewed), key=data\_right\_skewed.count)

# Plotting the graph

plt.hist(data\_right\_skewed, bins=10, edgecolor='black', color='lightgreen')

plt.axvline(x=mean\_right\_skewed, color='red', label='Mean')

plt.axvline(x=median\_right\_skewed, color='blue', label='Median')

plt.axvline(x=mode\_right\_skewed, color='orange', label='Mode')

plt.legend()

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.title('Right-Skewed Data')

plt.show()