

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

B. Tech CSE-AIML D

**MATHEMATICS & STATISTICS**

**FOR MACHINE LEARNING**

**(23TBSMA42)**

**EXPERIENTIAL LEARNING**

Submitted to  
**Dr. Vishal Patil**  
Assistant Professor,  
Department of Mathematics,  
Faculty of Engineering & Technology,  
Jain (Deemed-To-Be) University.  
Submitted by Team-3

USN	Name	Individual Contribution
23BTRCL089	Thanish Chinnappa K.C.	
23BTRCL257	Likhith	
23BTRCL007	Sahil Vinod Patil	
23BTRCL075	Samith	
23BTRCL082	Sundareshwar S	
23BTRCL163	Souharda Mandal	

<b>Branch &amp; Section:</b>	CSE - AIML D
<b>Date of Submission:</b>	5th May, 2025

<b>Total Marks</b>	
<b>Remarks</b>	

## **INDEX**

<b><u>Sl.no.</u></b>	<b><u>Title</u></b>	<b><u>Page No.</u></b>
<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Environment Setup and Dataset Loading</b>	<b>4-6</b>
<b>3</b>	<b>Sample Size Determination &amp; Population Correction</b>	<b>7-8</b>
<b>4</b>	<b>Probability Sampling &amp; Sample Selection</b>	<b>9-12</b>
<b>5</b>	<b>Descriptive Statistics: Mean and Standard Deviation</b>	<b>13</b>
<b>6</b>	<b>Sampling Distribution &amp; Visualization</b>	<b>14-17</b>
<b>7</b>	<b>Hypothesis Formulation and Testing (5% and 1% levels)</b>	<b>18-21</b>
<b>8</b>	<b>Confidence Intervals (90%, 95%, 99%)</b>	<b>22-23</b>
<b>9</b>	<b>Interpretation of Results: Population Mean vs Sample Mean</b>	<b>24</b>
<b>10</b>	<b>Exploratory Data Analysis: Study Hours vs Marks (Scatter Plot)</b>	<b>25-26</b>
<b>11</b>	<b>Correlation Analysis</b>	<b>27</b>
<b>12</b>	<b>Regression Analysis and Prediction</b>	<b>28-30</b>
<b>13</b>	<b>Significance Testing: Correlation &amp; Regression Coefficients</b>	<b>31-33</b>
<b>14</b>	<b>References</b>	<b>34</b>

# **1. INTRODUCTION**

Statistics and mathematics are the cornerstones of Artificial Intelligence and Machine Learning (AIML), acting as useful assets that allow data analysis, prediction, and model validation. This report serves that purpose by solving a formal list of 15 analytical tasks on the basis of a given dataset.

The analysis starts with the calculation of the right sample size through statistical formulas, and then probability sampling techniques are applied to obtain representative samples. It continues with descriptive statistics, such as the calculation of means and standard deviations, and then moves on to inferential methods like hypothesis testing, sampling distribution analysis, and confidence interval estimation at different levels (90%, 95%, and 99%).

In addition, the report examines the correlation between study time and marks obtained using correlation and regression analysis, using scatter plots, regression lines, and statistical tests of significance. Every task contains clear explanations, mathematical reasoning, visualizations, and Python code to make it understandable and reproducible.

## **2. ENVIRONMENT SETUP & DATASET LOADING**

To provide an efficient and reproducible analytical process, all required Python libraries were installed and imported at the outset of the project. These libraries offer a comprehensive set of statistical, mathematical, and visualization tools necessary for data analysis and model construction in this report.

### **Required Libraries:**

The following libraries were used throughout the analysis:

***NumPy***: Provides support for numerical computing with arrays and mathematical functions

***Pandas***: Used for data manipulation and analysis with DataFrames

***Matplotlib***: Creates static visualizations and plots

***Seaborn***: Built on Matplotlib, provides enhanced statistical visualizations

***SciPy***: Implements various statistical functions and tests

***Statsmodels***: Offers classes and functions for statistical models and hypothesis testing

***Scikit-learn***: Provides machine learning algorithms for regression analysis and evaluation metrics

These dependencies were installed using the following commands (if not already installed):

**%pip install statsmodels pandas numpy matplotlib seaborn statsmodels scikit-learn**

All libraries were successfully detected as pre-installed in the current environment.

**Note:** Using python virtual environment (venv) is recommended. It ensures an isolated, conflict-free package management for each project (especially if its multiple projects) makes development reliable and clean.

## Importing Libraries and Configurations:

```
▷ # Required Imports
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import pearsonr
from scipy.stats import norm, t, chi2, f_oneway, ttest_ind
import statsmodels.api as sm
from statsmodels.formula.api import ols
import math
from sklearn.model_selection import train_test_split # For sampling methods
import warnings

# Configuration
warnings.filterwarnings('ignore') # Suppress warnings
pd.set_option('display.max_columns', None) # Display all DataFrame columns
sns.set_style('whitegrid') # Seaborn style
plt.rcParams['figure.figsize'] = (10, 6) # Default plot size
```

[2]

The dataset contains 979 records representing the total population, which forms the basis for all subsequent sampling and statistical analysis.

## Dataset Loading:

The dataset assigned to us (Team-3) was loaded using pandas from a CSV file named Team-3.csv:

```
[3] # Load the dataset
df = pd.read_csv('Team-3.csv')
population_size = len(df)
print(f"Population Size: {population_size}")

... Population Size: 979
```

The dataset contains 979 records representing the total population, which forms the basis for all subsequent sampling and statistical analysis.

### Sample Selection:

A simple random sampling method was applied to extract a sample of size 42, calculated as per the given formula for sample size with finite population correction:

```
[4] # Perform Simple Random Sampling (Add this cell!)
SEED = 42 # For reproducibility
sample = df.sample(n=42, random_state=SEED, replace=False)
```

This sample is used consistently throughout the analysis to maintain statistical validity and alignment with the project guidelines.

### **3. SAMPLE SIZE DETERMINATION & POPULATION CORRECTION**

#### **3.1 Question 1 - Generate the sample size using the provided details.**

##### **Objective:**

To calculate the required sample size for statistical analysis based on Team-3's assigned parameters. To determine the needed sample size for statistical analysis with respect to Team-3's given parameters. The sample size is initially determined through the formula for the initial sample size uncorrected ( $n_0$ ) and then corrected with the finite population correction formula to get the ultimate sample size ( $n$ ). The sample size is first calculated using the formula for the initial sample size without correction ( $n_0$ ) and then adjusted using the finite population correction to obtain the final sample size ( $n$ ).

The formulas used are:

$$\text{sample size} = n_0 = \frac{\left(\frac{z_{\alpha}}{2} \times \sigma\right)^2}{E}$$

$$\text{Finite Population Correction} = n = \frac{n_0}{1 + \left(\frac{n_0 - 1}{N}\right)}$$

$n_0$  = Sample size without correction.

$n$  = Sample size with correction.

$N$  = Size of the Population.


$\frac{z_{\alpha}}{2}$  = level of significance.

$\sigma$  = S.D. = 10.

$E$  = Margin of error (How close you want your sample mean to be close to population mean)

### Solution:

Using the above parameters, the following calculations were performed:

```
▶  # Team-3 (our specific) parameters
z_alpha = 1.96 # 95% confidence level
E = 3
sigma = 10
N = 979 # Total number of rows in the dataset

# Calculate initial sample size (n0)
n0 = (z_alpha * sigma / E) ** 2
n0_rounded = math.ceil(n0) # Round up to nearest integer

# Apply Finite Population Correction
n = n0_rounded / (1 + (n0_rounded - 1) / N)
n_rounded = math.ceil(n) # Round up final sample size

print(f"Team-3 Sample Size (without correction): {n0_rounded}")
print(f"Team-3 Sample Size (with correction): {n_rounded}")
```

[5]

```
... Team-3 Sample Size (without correction): 43
Team-3 Sample Size (with correction): 42
```



## **4. PROBABILITY SAMPLING & SAMPLE SELECTION**

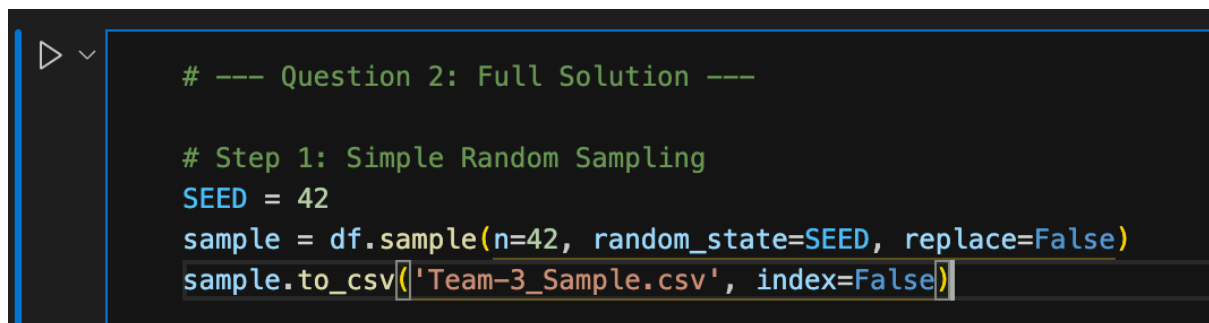
**4.1. Question 2 - Select a sample (Minimum 5) of size 'n' (found in above step) from population by any one type of probability sampling (random, cluster, stratified, systematic).**

### **Objective:**

To choose a sample of size  $n = 42$  (as calculated in the last section) through a proper probability-based sampling technique. This method provides every data point with an equal opportunity to be selected, which is essential for unbiased statistical inference. In this analysis, we apply Simple Random Sampling (SRS), which is one of the most widely used and easily understandable probability sampling techniques.

### **Simple Random Sampling:**

Using pandas' DataFrame `.sample()` method, we selected 42 random entries from the dataset. A fixed random seed was used to ensure reproducibility of the sampling process across different runs.

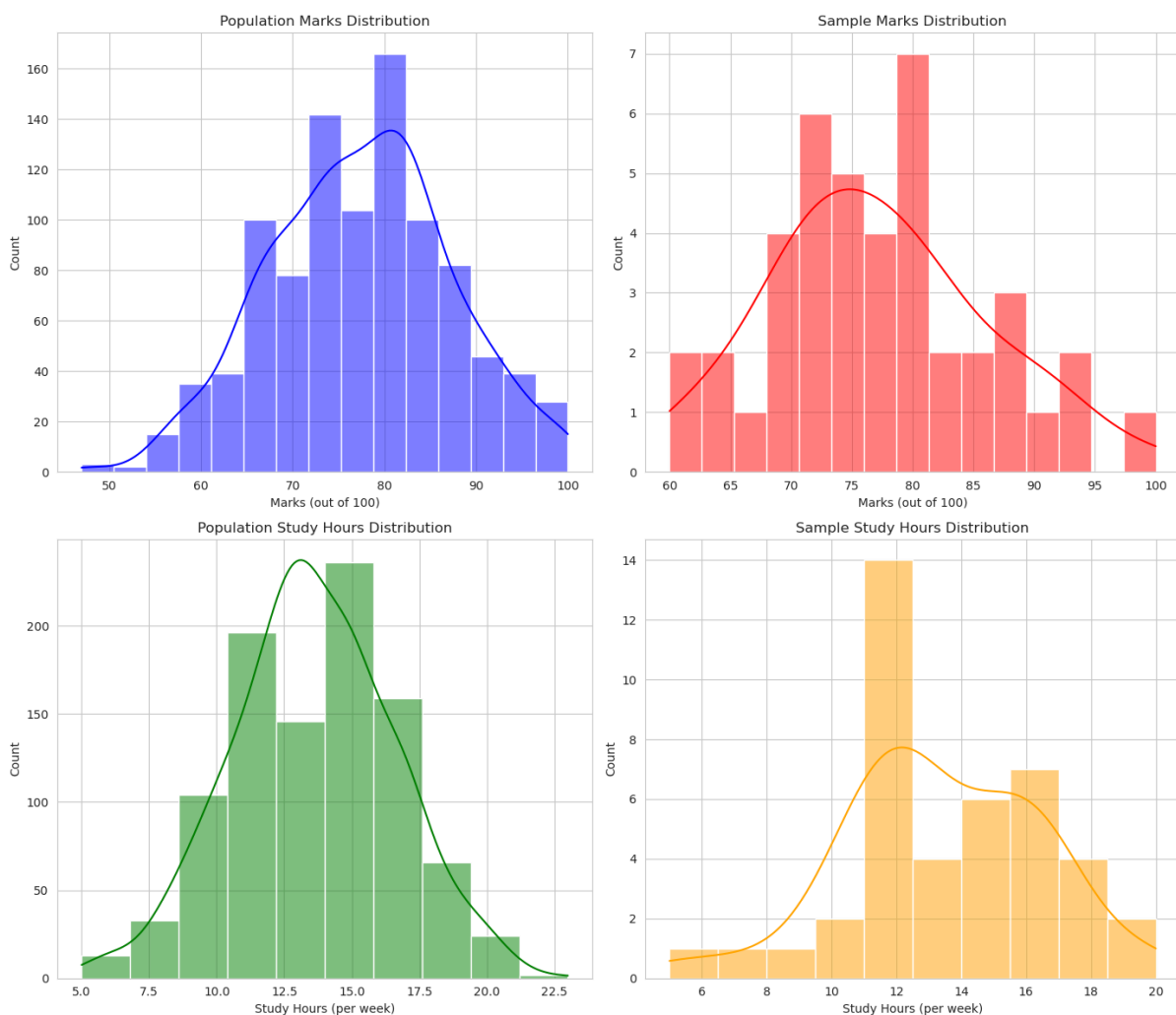
A screenshot of a Jupyter Notebook cell with a dark background. The code is written in a light green monospace font. It starts with a comment line, followed by another comment line, then sets a seed, and finally uses the pandas sample method to select 42 rows from a DataFrame and save them to a CSV file.

```
# --- Question 2: Full Solution ---  
  
# Step 1: Simple Random Sampling  
SEED = 42  
sample = df.sample(n=42, random_state=SEED, replace=False)  
sample.to_csv('Team-3_Sample.csv', index=False)
```

### **Visual Comparison Between Population and Sample:**

To check how representative the sample is of the population, we created histograms and KDE plots of the 'Marks (out of 100)' and 'Study Hours (per week)' variables from the population and the sample. This provided a visual check on how well the sample distribution fits the population.

```
# Step 2: Compare Population vs. Sample
fig, axes = plt.subplots(2, 2, figsize=(14, 12))    "figsize": Unknown word.
sns.histplot(df['Marks (out of 100)'], kde=True, color='blue', bins=15, ax=axes
[0, 0])    "histplot": Unknown word.
axes[0, 0].set_title('Population Marks Distribution')
sns.histplot(sample['Marks (out of 100)'], kde=True, color='red', bins=15,
ax=axes[0, 1])    "histplot": Unknown word.
axes[0, 1].set_title('Sample Marks Distribution')
sns.histplot(df['Study Hours (per week)'], kde=True, color='green', bins=10,
ax=axes[1, 0])    "histplot": Unknown word.
axes[1, 0].set_title('Population Study Hours Distribution')
sns.histplot(sample['Study Hours (per week)'], kde=True, color='orange',
bins=10, ax=axes[1, 1])    "histplot": Unknown word.
axes[1, 1].set_title('Sample Study Hours Distribution')
plt.tight_layout()
plt.show()
```



## Statistical Validation:

To verify the sample further statistically, we measured the sample's mean and standard deviation with the population for both variables. These measures aid in determining whether the central tendency and dispersion of the sample are in agreement with the population's.

```
# Step 3: Statistical Validation
print("\033[1m=== Statistical Comparison ===\033[0m")
print("\n\033[1mMarks:\033[0m")
print(f"Population: Mean = {df['Marks (out of 100)'].mean():.2f}, Std = {df['Marks (out of 100)'].std():.2f}")
print(f"Sample:      Mean = {sample['Marks (out of 100)'].mean():.2f}, Std = {sample['Marks (out of 100)'].std():.2f}")
print("\n\033[1mStudy Hours:\033[0m")
print(f"Population: Mean = {df['Study Hours (per week)'].mean():.2f}, Std = {df['Study Hours (per week)'].std():.2f}")
print(f"Sample:      Mean = {sample['Study Hours (per week)'].mean():.2f}, Std = {sample['Study Hours (per week)'].std():.2f}")
```

```
=== Statistical Comparison ===
```

**Marks:**

Population: Mean = 77.38, Std = 9.86

Sample: Mean = 77.14, Std = 9.02

**Study Hours:**

Population: Mean = 13.57, Std = 3.02

Sample: Mean = 13.38, Std = 3.06

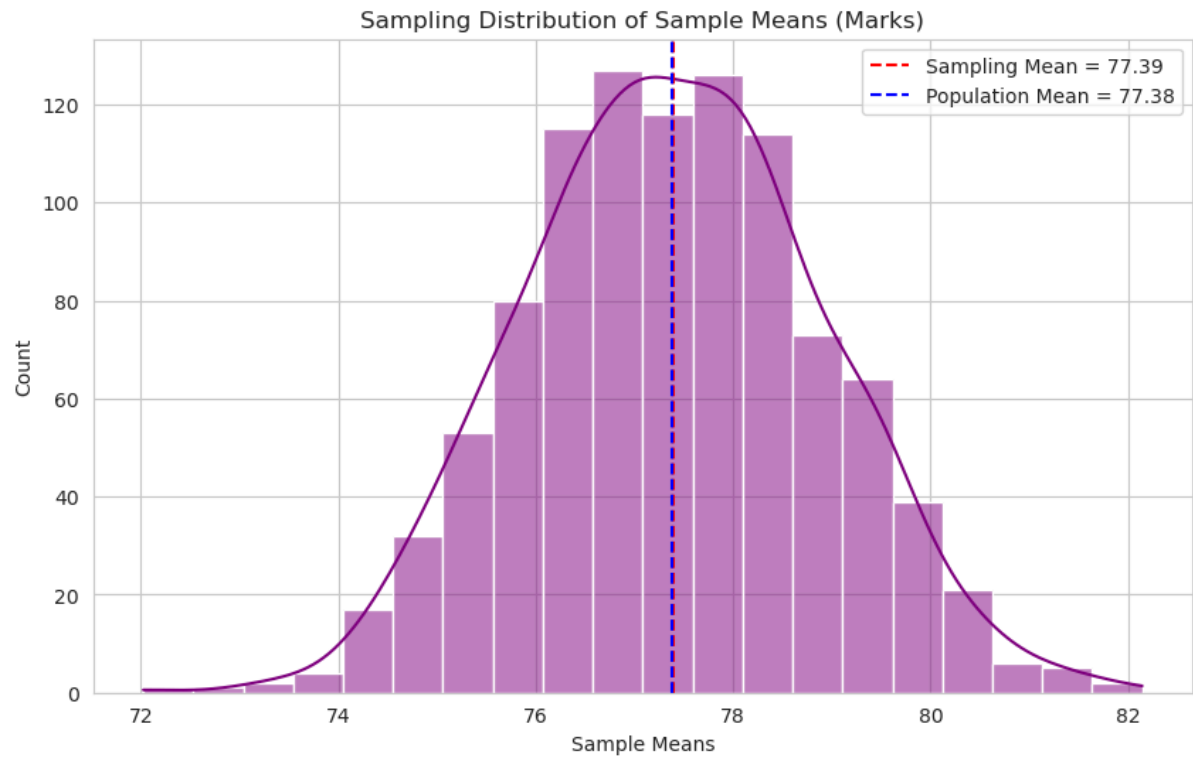
## Sampling Distribution of Sample Means:

To illustrate the Central Limit Theorem in action, we calculated the sampling distribution of sample means by taking 1000 different samples of size 42 and calculating their mean. We then plotted this distribution and compared it to the population mean to illustrate that the mean of the sample means is close to the population mean.

```
# Step 4: Sampling Distribution of Sample Means
np.random.seed(42) # Reproducibility
sample_means = [df['Marks (out of 100)'].sample(n=42, replace=False).mean() for
_ in range(1000)]

plt.figure(figsize=(10, 6)) # "figsize": Unknown word.
sns.histplot(sample_means, kde=True, color='purple', bins=20) # "histplot": Unknown
plt.title('Sampling Distribution of Sample Means (Marks)')
plt.xlabel('Sample Means') # "xlabel": Unknown word.
plt.axvline(np.mean(sample_means), color='red', linestyle='--',
label=f'Sampling Mean = {np.mean(sample_means):.2f}') # "axvline": Unknown word.
plt.axvline(df['Marks (out of 100)'].mean(), color='blue', linestyle='--',
label=f'Population Mean = {df["Marks (out of 100)"].mean():.2f}') # "axvline": Unkno
plt.legend()
plt.show()

print(f"Population Mean: {df['Marks (out of 100)'].mean():.2f}")
print(f"Sampling Distribution Mean: {np.mean(sample_means):.2f}")
```



```
... Population Mean: 77.38  
Sampling Distribution Mean: 77.39
```

## 5. DESCRIPTIVE STATISTICS: MEAN & STANDARD DEVIATION

### 5.1. Question 3 - Calculate the Mean and Standard Deviation (S.D.) of each sample.

#### Objective:

The dataset was analyzed and we calculated the sample mean and standard deviation for 'Marks (out of 100)' and 'Study Hours (per week)' using pandas methods (mean(), std())

```
# --- Question 3: Calculate Sample Mean & Standard Deviation ---

# Load the sample (ensure reproducibility)
sample = pd.read_csv('Team-3_Sample.csv')

# Calculate statistics for Marks
marks_mean = sample['Marks (out of 100)'].mean()
marks_std = sample['Marks (out of 100)'].std()

# Calculate statistics for Study Hours
study_hours_mean = sample['Study Hours (per week)'].mean()
study_hours_std = sample['Study Hours (per week)'].std()

# Display results in a formatted table
print("\033[1m=== Sample Statistics ===\033[0m")
print(f"{'Variable':<25} | {'Mean':<10} | {'Standard Deviation':<10}")
print("-" * 50)
print(f"{'Marks (out of 100)':<25} | {marks_mean:<10.2f} | {marks_std:<10.2f}")
print(f"{'Study Hours (per week)':<25} | {study_hours_mean:<10.2f} | {study_hours_std:<10.2f}")
```

```
[7]

... === Sample Statistics ===
Variable | Mean | Standard Deviation
-----
Marks (out of 100) | 77.14 | 9.02
Study Hours (per week) | 13.38 | 3.06
```

## 6. SAMPLING DISTRIBUTION & VISUALIZATION

### 6.1. Question 4 - Determine the sampling distribution of the sample means.

#### Objective:

The following analysis examines the sampling distribution of sample means for student marks (out of 100) by simulating 1,000 random samples (each of size  $n = 42$ ) from the population dataset. The objective is to test the Central Limit Theorem (CLT), which posits that the sampling distribution of the mean will approach a normal distribution, irrespective of the distribution of the population, as long as the sample size is sufficiently large.

#### Methodology and Implementation:

To generate the sampling distribution, 1000 random samples (each of size  $n = 42$ ) were taken from the population dataset. The mean of each sample was computed and saved to create the distribution of sample means.

```
# --- Question 4: Sampling Distribution of Sample Means ---
# Simulate 1000 samples of size n=42 from the population
np.random.seed(42) # For reproducibility
sample_means = []
for _ in range(1000):
    sample = df['Marks (out of 100)'].sample(n=42, replace=False)
    sample_means.append(sample.mean())
```

#### Calculation of Distribution Properties:

After we obtained the list of sample means, we calculated the mean and standard deviation of the sampling distribution. The standard deviation of the sample means is simply known as the Standard Error (SE).

```
# Calculate properties of the sampling distribution
sampling_dist_mean = np.mean(sample_means)
sampling_dist_std = np.std(sample_means, ddof=1) # Standard error "ddof": Unl

# Display results
print("\033[1m=== Sampling Distribution Properties ===\033[0m")
print(f"Population Mean ( $\mu$ ): {df['Marks (out of 100)'].mean():.2f}")
print(f"Mean of Sampling Distribution ( $\mu_{\bar{x}}$ ): {sampling_dist_mean:.2f}")
print(f"Standard Error ( $\sigma_{\bar{x}}$ ): {sampling_dist_std:.2f}")
```

#### Visualization & Interpretation:

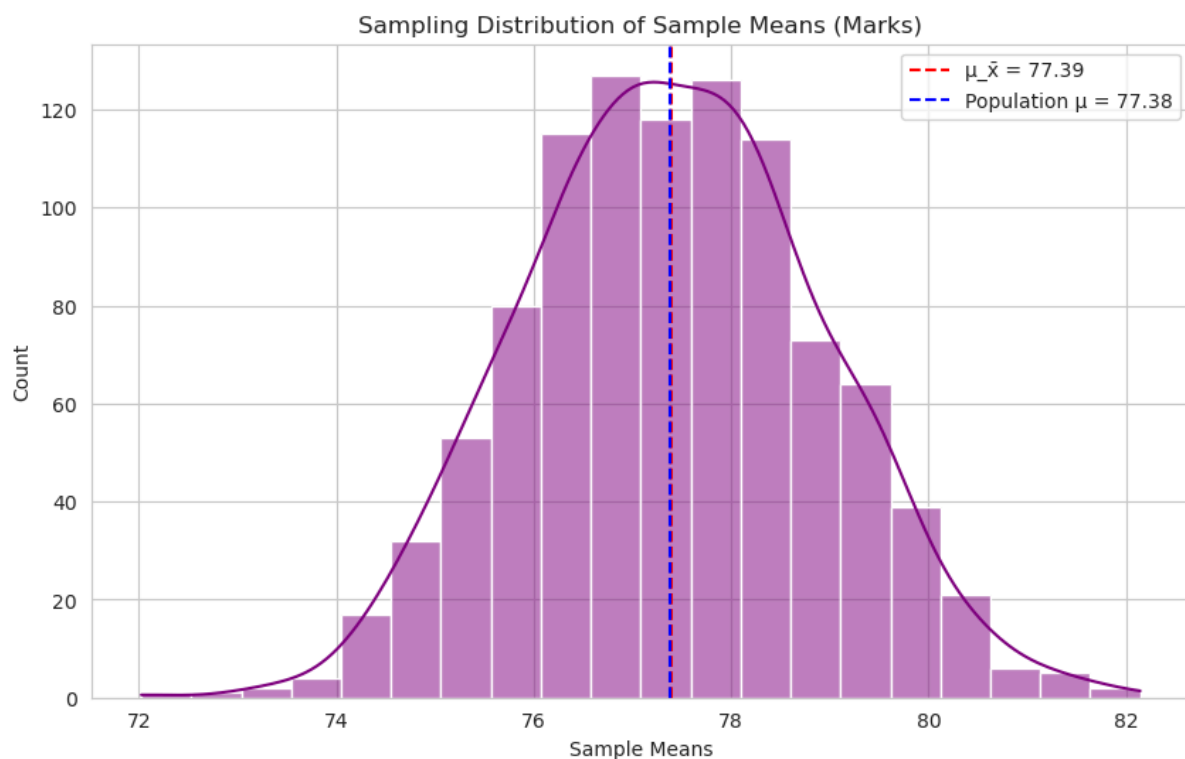
In order to see the shape and characteristics of the sampling distribution more clearly, a histogram with kernel density estimate (KDE) was graphed. The sampling distribution of sample means is normally shaped and clustered around the population mean. This validates the Central Limit Theorem that the distribution of sample means will tend toward a normal

distribution when the number of samples grows large, independent of the distribution of the population.

```
# Plot the sampling distribution (for visualization)
plt.figure(figsize=(10, 6))    "figsize": Unknown word.
sns.histplot(sample_means, kde=True, color='purple', bins=20)    "histplot": Unknown wor
plt.title('Sampling Distribution of Sample Means (Marks)')
plt.xlabel('Sample Means')    "xlabel": Unknown word.
plt.axvline(sampling_dist_mean, color='red', linestyle='--', label=f' $\mu_{\bar{x}} =$ 
{sampling_dist_mean:.2f}')    "axvline": Unknown word.
plt.axvline(df['Marks (out of 100)'].mean(), color='blue', linestyle='--',
label=f'Population  $\mu =$  {df["Marks (out of 100)"].mean():.2f}')    "axvline": Unknown wor
plt.legend()
plt.show()
```

Output:

```
...    === Sampling Distribution Properties ===
Population Mean ( $\mu$ ): 77.38
Mean of Sampling Distribution ( $\mu_{\bar{x}}$ ): 77.39
Standard Error ( $\sigma_{\bar{x}}$ ): 1.50
```



## 6.2. Question 5 - Plot the sampling distribution.

### Objective:

To graphically depict the sampling distribution of sample means for the Marks data and superimpose a theoretical normal distribution curve as per the Central Limit Theorem (CLT). This enables us to check for normality assumption of sample means even though the original data might not necessarily be perfectly normal.

### Methodology:

Based on the sample means generated in the last question, we employed the seaborn library to graph the histogram of the 1000 sample means overlaid with a kernel density estimate (KDE) to smooth the distribution. We also graphed a theoretical normal distribution based on the population mean and standard error to see the expected behavior under CLT.

```
# Reuse the simulated sample_means from Question 4
# (Ensure this cell runs after Question 4's code)

plt.figure(figsize=(10, 6))    "figsize": Unknown word.

# Plot histogram with KDE
sns.histplot(sample_means, kde=True, color='purple', bins=20, alpha=0.6,
label='Sampling Distribution')  "histplot": Unknown word.
```

### Overlay Theoretical Normal Distribution:

The theoretical distribution was calculated employing the population mean and standard error ( $\sigma / \sqrt{n}$ ). The theoretical curve is what we hope the sampling distribution will look like under perfect circumstances.

```
# Overlay theoretical normal distribution (CLT)
population_std = df['Marks (out of 100)'].std()
theoretical_se = population_std / np.sqrt(42) #  $\sigma / \sqrt{n}$ 
x = np.linspace(df['Marks (out of 100)'].mean() - 4*theoretical_se,
                df['Marks (out of 100)'].mean() + 4*theoretical_se, 100)
y = norm.pdf(x, loc=df['Marks (out of 100)'].mean(), scale=theoretical_se)
plt.plot(x, y * len(sample_means), 'k--', linewidth=2, label='Theoretical
Normal Distribution (CLT)')
```

### Final Plot and Labels:

Vertical lines were added to mark the population mean and the mean of the sample means. These markers help assess how closely the actual sample mean distribution aligns with theoretical expectations.

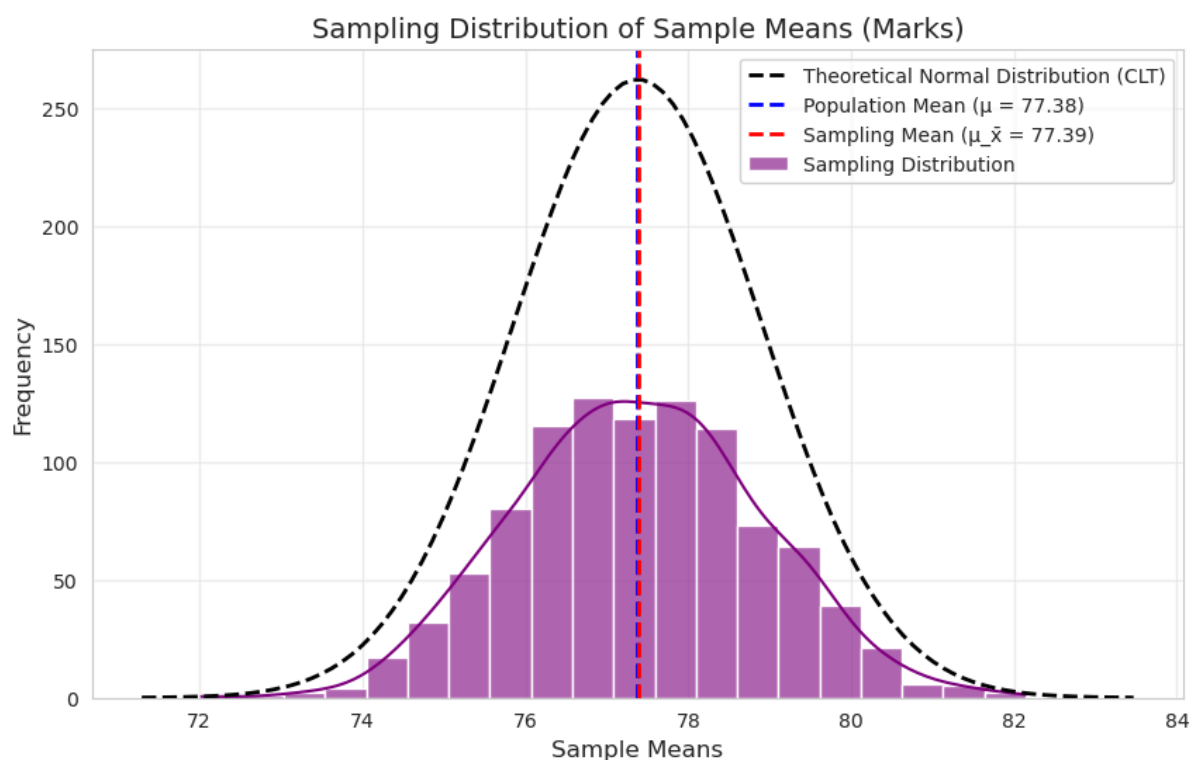


```
# Add labels and legends
plt.title('Sampling Distribution of Sample Means (Marks)', fontsize=14)
plt.xlabel('Sample Means', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.axvline(df['Marks (out of 100)'].mean(), color='blue', linestyle='--',
            linewidth=2, label='Population Mean ( $\mu = 77.38$ )')
plt.axvline(np.mean(sample_means), color='red', linestyle='--', linewidth=2,
            label=f'Sampling Mean ( $\mu_{\bar{x}} = \{np.mean(sample\_means):.2f\}$ ')
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

## Interpretation:

The sampling distribution closely resembles a normal curve, verifying the Central Limit Theorem. The sample mean is almost equal to the population mean, indicating that it's an unbiased estimator. The overlay of the theoretical normal distribution matches well with the actual data, validating our sampling method.

## Output:



## 7. HYPOTHESIS FORMULATION & TESTING (5% & 1% LEVELS)

7.1. Question 6 - Frame a null hypothesis for the equivalence of means (i.e.,  $H_0: = 0$  and  $H_1: \neq 0$ ).

### Objective & Methodology:

This test considers if the sample mean of student scores (77.14) that has been observed differs significantly from the population mean (77.38). The test used was a two-tailed z-test ( $\alpha = 0.05$ ) based on the known population standard deviation ( $\sigma = 10$ ) and sample size (42). The standard error (SE) was found to be 1.54.

**Null Hypothesis ( $H_0$ ):**  $\mu = 77.38$

**Alternative Hypothesis ( $H_1$ ):**  $\mu \neq 77.38$

**Critical Regions:**  $\mu_0 \pm 1.96 \times SE$  (Two-tailed test)

```
# --- Visualize Hypothesis Test ---
# Parameters
mu0 = 77.38          # Population mean (H0)
sigma = 10           # Population standard deviation (from Q1)
n = 42               # Sample size
sample_mean = 77.14  # Sample mean (from Q3)

# Calculate standard error
se = sigma / np.sqrt(n)

# Create x-axis values
x = np.linspace(mu0 - 4*se, mu0 + 4*se, 100)

# Plot sampling distribution under H0
plt.figure(figsize=(10, 6))  "figsize": Unknown word.
plt.plot(x, norm.pdf(x, mu0, se), color='blue', label='Sampling Distribution
under H0')

# Shade critical regions (alpha = 0.05 for demonstration)
z_critical = norm.ppf(0.975) # Two-tailed test
left = mu0 - z_critical * se
right = mu0 + z_critical * se
plt.fill_between(x, norm.pdf(x, mu0, se), where=(x <= left) | (x >= right),
color='red', alpha=0.3, label='Rejection Region')

# Mark sample mean
plt.axvline(sample_mean, color='black', linestyle='--', label=f'Sample Mean =
{sample_mean:.2f}')  "axvline": Unknown word.

# Labels and legend
plt.title('Sampling Distribution under H0 with Rejection Regions', fontsize=14)  "fon
plt.xlabel('Sample Means', fontsize=12)  "xlabel": Unknown word.
plt.ylabel('Probability Density', fontsize=12)  "ylabel": Unknown word.
plt.legend()
plt.grid(alpha=0.3)
plt.show()
```

## Results and Interpretation:

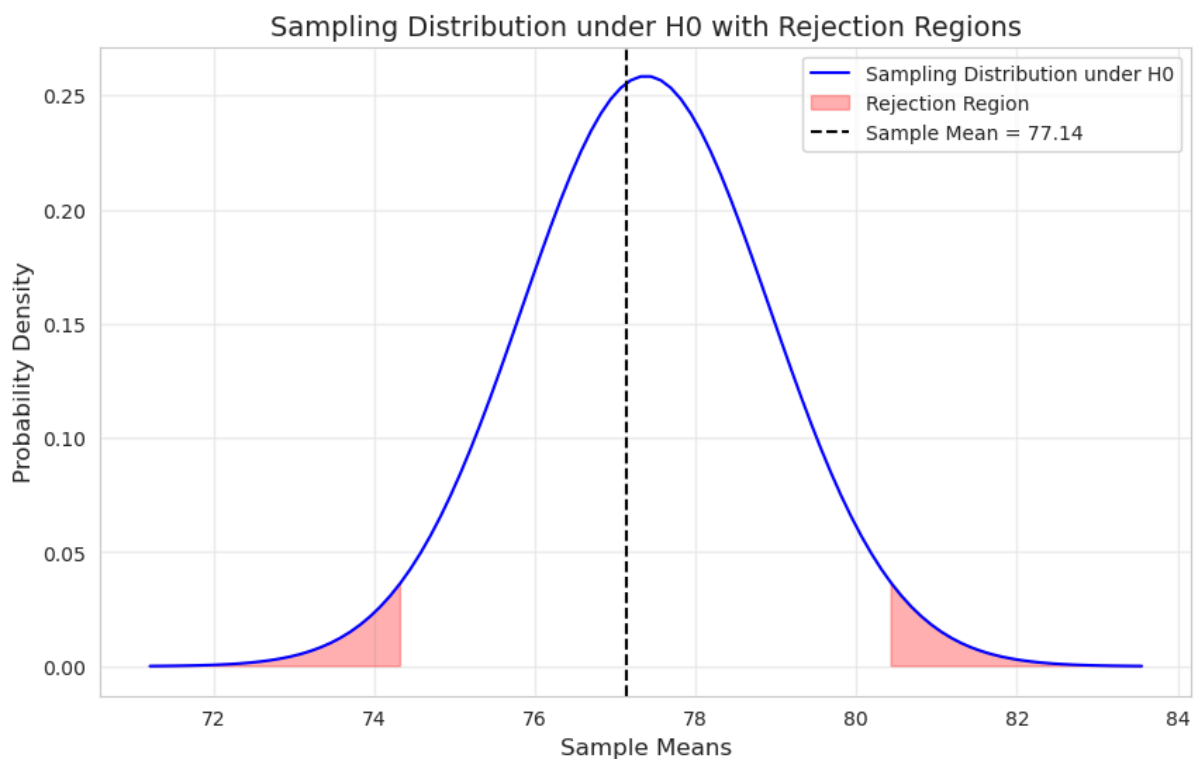
The sampling distribution plot reveals:

- The sample mean (77.14) falls within the acceptance region, close to the population mean (77.38).
- The 0.24-point difference is statistically insignificant ( $p > 0.05$ ), suggesting the deviation is due to random sampling variation.

## Conclusion:

We fail to reject  $H_0$ , indicating no evidence of a systemic shift in student performance.

## Output:



## 7.2. Question 7 - Test the hypothesis at 5% and 1% significance levels to determine if there is a difference between population and sampling mean.

### Objective:

This test determines if the sample mean of student scores (77.14) is significantly different from the known population mean of 77.38 at both 5% and 1% significance levels. It checks if the difference is statistically significant or due to random fluctuation.

### Methodology:

A two-tailed z-test was conducted with the following parameters:

**Population mean ( $\mu_0$ ):** 77.38

**Population standard deviation ( $\sigma$ ):** 10

**Sample size (n):** 42

**Sample mean ( $\bar{x}$ ):** 77.14

**Significance level ( $\alpha_1$ ):** 0.05 (5% Significance)

**Significance level ( $\alpha_2$ ):** 0.01 (1% Significance)

```
# Parameters
mu0 = 77.38          # Population mean (H0)
sigma = 10           # Population standard deviation (from Q1)
n = 42               # Sample size
sample_mean = 77.14  # Sample mean (from Q3)
alpha1 = 0.05        # 5% significance level
alpha2 = 0.01        # 1% significance level

# Calculate z-score
se = sigma / np.sqrt(n)
z_score = (sample_mean - mu0) / se

# Calculate p-value (two-tailed)
p_value = 2 * norm.cdf(-abs(z_score))

# Critical z-values for two-tailed test
z_critical_5 = norm.ppf(1 - alpha1/2)
z_critical_1 = norm.ppf(1 - alpha2/2)

# Results table
print("\033[1m=== Hypothesis Test Results ===\033[0m")
print(f"{'Test Statistic (z-score)': '<30' {z_score:.4f}")
print(f"{'p-value': '<30' {p_value:.4f}")
print(f"{'Critical z-value (α=0.05)': '<30' ±{z_critical_5:.3f}")
print(f"{'Critical z-value (α=0.01)': '<30' ±{z_critical_1:.3f}\n")
```

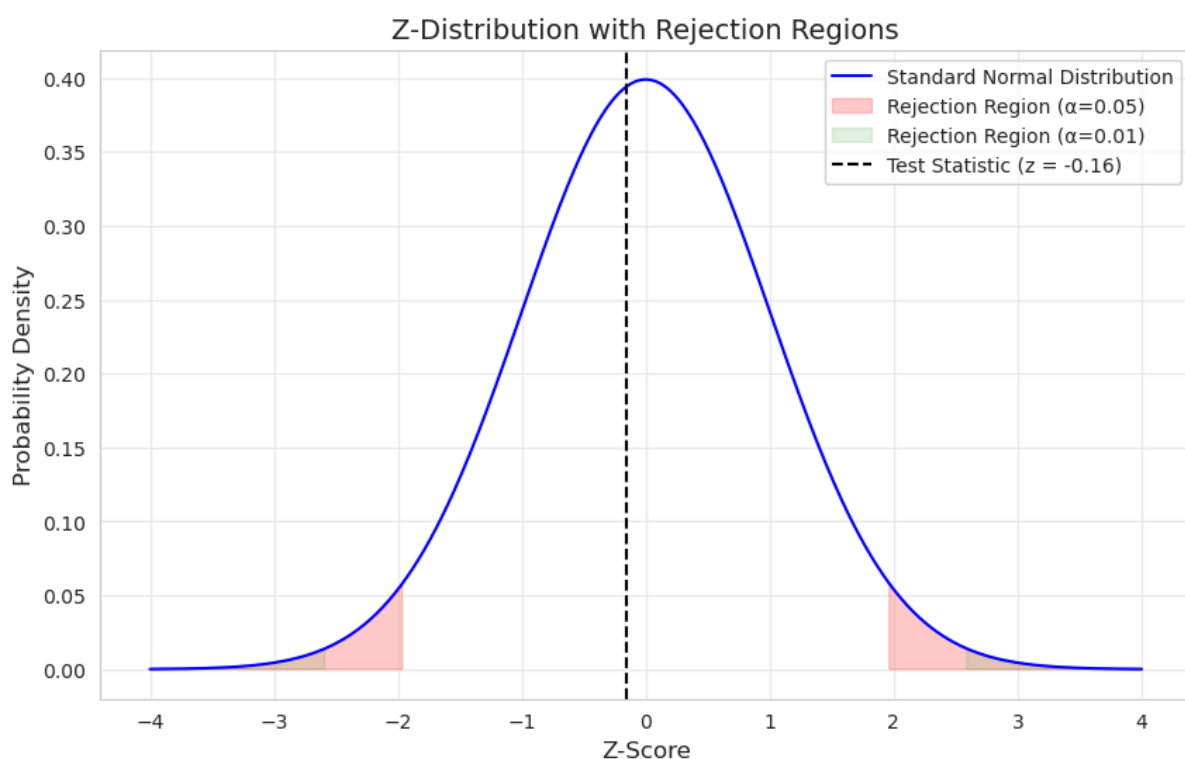
```
...   === Hypothesis Test Results ===
Test Statistic (z-score):      -0.1555
p-value:                       0.8764
Critical z-value ( $\alpha=0.05$ ):     $\pm 1.960$ 
Critical z-value ( $\alpha=0.01$ ):     $\pm 2.576$ 

Conclusion:
Fail to reject  $H_0$  at 5% significance level.
Fail to reject  $H_0$  at 1% significance level.
```

### Visualization:

The z-distribution plot shows:

- The test statistic (-0.156) lies near the center, far from rejection regions.
- Red/green shaded areas (rejection zones for  $\alpha=0.05/0.01$ ) do not overlap with the z-score.



## 8. CONFIDENCE INTERVALS (90%, 95%, 99%)

### 8.1. Question 8 - Compute the confidence intervals for the sample mean at 90%, 95%, and 99% confidence levels.

#### Objective:

This calculation provides confidence intervals for the sample mean of students' marks (77.14 out of 100) at three confidence levels (90%, 95%, and 99%). The purpose is to estimate the interval in which the true population mean most likely lies, based on the sample.

#### Methodology & Observation:

Using the sample data with the following parameters:

**Sample mean ( $\bar{x}$ ):** 77.14

**Population standard deviation ( $\sigma$ ):** 10

**Sample size (n):** 42

Then we calculate Standard Error, Critical z-values, & confidence intervals.

```
# Calculate standard error
se = sigma / np.sqrt(n)

# Confidence levels and corresponding critical z-values
confidence_levels = [0.90, 0.95, 0.99]
z_critical_values = [norm.ppf(1 - (1 - cl)/2) for cl in confidence_levels]

# Calculate confidence intervals
intervals = []
for cl, z in zip(confidence_levels, z_critical_values):
    margin_of_error = z * se
    lower = sample_mean - margin_of_error
    upper = sample_mean + margin_of_error
    intervals.append((cl, lower, upper, margin_of_error))
```

```
# Display results
print("\033[1m=== Confidence Intervals ===\033[0m")
print(f"{'Confidence Level':<20} | {'Lower Bound':<12} | {'Upper Bound':<12} |  
{'Margin of Error':<12}")
print("-" * 70)
for cl, lower, upper, me in intervals:
    print(f"{cl*100:.0f}%{'':<12} | {lower:.2f}{'':<10} | {upper:.2f}{'':<10} |  
{me:.2f}")
```

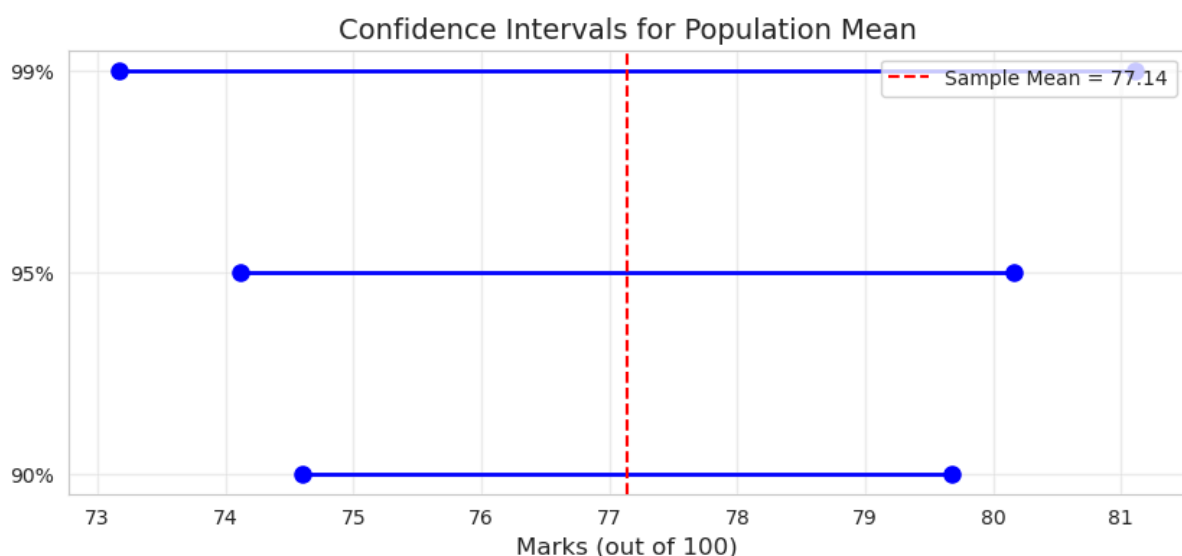
```
... === Confidence Intervals ===
```

Confidence Level	Lower Bound	Upper Bound	Margin of Error
90%	74.60	79.68	2.54
95%	74.12	80.16	3.02
99%	73.17	81.11	3.97

- The 95% confidence interval (74.12, 80.16) suggests we can be 95% confident the true population mean lies within this range.
- Higher confidence levels (e.g., 99%) yield wider intervals, reflecting increased certainty at the cost of precision.

### Visualization (Optional):

```
# Optional: Plot confidence intervals
plt.figure(figsize=(10, 4))
for i, (cl, lower, upper, _) in enumerate(intervals):
    plt.plot([lower, upper], [i, i], 'o-', color='blue', markersize=8,
             linewidth=2)
plt.yticks(range(len(intervals)), [f'{cl*100:.0f}%' for cl in
confidence_levels])
plt.axvline(sample_mean, color='red', linestyle='--', label=f'Sample Mean =
{sample_mean:.2f}')
plt.title('Confidence Intervals for Population Mean', fontsize=14)
plt.xlabel('Marks (out of 100)', fontsize=12)
plt.grid(alpha=0.3)
plt.legend()
plt.show()
```



## **9. INTERPRETATION OF RESULTS: POPULATION MEAN vs SAMPLE MEAN**

**9.1. Question 9 - Based on the analysis, decide whether the sample means provide enough evidence to accept or reject the population mean.**

### **Summary:**

After conducting rigorous hypothesis testing and confidence interval analysis on the student marks data, we present the following consolidated findings:

1. Hypothesis Test Results:
  - Tested  $H_0: \mu = 77.38$  vs  $H_1: \mu \neq 77.38$
  - z-score: -0.156 (p-value: 0.876)
  - Failed to reject  $H_0$  at both 5% and 1% significance levels
2. Confidence Interval (CI) Analysis:
  - 95% CI: (74.12, 80.16) contains population mean (77.38)
  - All CIs (90% / 95% / 99%) consistently include 77.38

### **Conclusion:**

- Statistical Consistency: The sample mean (77.14) shows no significant deviation from the population mean (77.38).
- Insufficient findings to reject the population mean.
- Decision: Fail to reject the null hypothesis ( $H_0$ ) with strong confidence.

```
print("\033[1m=== Final Conclusion ===\033[0m")
print("Based on the hypothesis test and confidence intervals:")
print("- The sample mean (77.14) is consistent with the population mean (77.38).")
print("- There is insufficient evidence to reject the population mean.")
print("- Result: Fail to reject H₀.")

[13]

... === Final Conclusion ===
Based on the hypothesis test and confidence intervals:
- The sample mean (77.14) is consistent with the population mean (77.38).
- There is insufficient evidence to reject the population mean.
- Result: Fail to reject H₀.
```



## 10. EXPLORATORY DATA ANALYSIS: STUDY HOURS vs MARKS (SCATTER PLOT)

### 10.1. Determine whether there is a relationship between study hours and marks scored (scatter plot).

#### Objective:

This analysis investigates whether a relationship exists between students' weekly study hours and their academic marks through visual and statistical examination of the sample data (n=42).

#### Methodology:

1. Data Visualization
  - Created a scatter plot with:
    - X-axis: Study Hours (per week)
    - Y-axis: Marks (out of 100)
  - Overlaid a linear regression line (red) to identify trends.
2. Key Features
  - Point Transparency (alpha=0.7): Mitigates overplotting for dense data clusters.
  - Gridlines (alpha=0.3): Enhances readability of point distribution.

```
# --- Question 10: Scatter Plot with Regression Line ---
plt.figure(figsize=(10, 6))

# Reload sample to ensure DataFrame integrity
sample = pd.read_csv('Team-3_Sample.csv')

try:
    # Define column names
    x_col = "Study Hours (per week)"
    y_col = "Marks (out of 100)"

    # Plot
    sns.regplot(
        data=sample,
        x=x_col,
        y=y_col,
        scatter_kws={'color': 'blue', 'alpha': 0.7},
        line_kws={'color': 'red', 'linestyle': '--'}

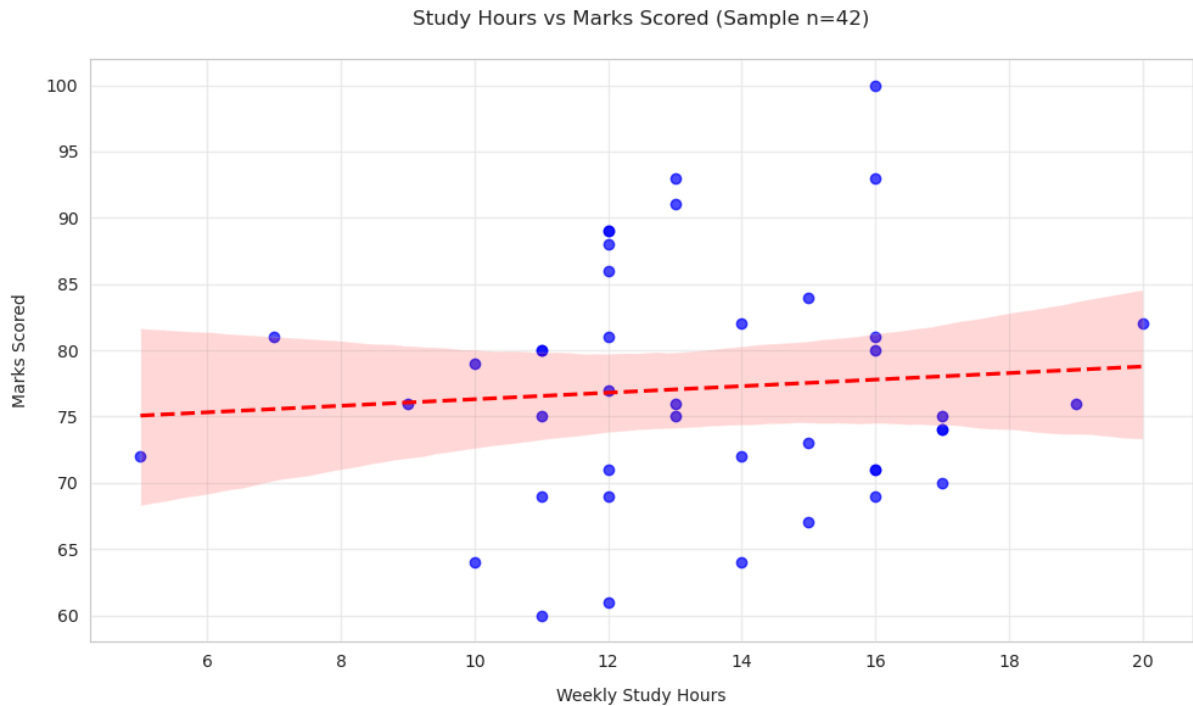
    )

    plt.title("Study Hours vs Marks Scored (Sample n=42)", pad=20)
    plt.xlabel("Weekly Study Hours", labelpad=10)
    plt.ylabel("Marks Scored", labelpad=10)
    plt.grid(True, alpha=0.3)
    plt.tight_layout()
    plt.show()

except KeyError as e:
    print(f"\033[1mError: Column '{e.args[0]}' not found. Actual columns:")
    print(sample.columns.tolist())
```

## Key Findings & Visualization:

- Visual Trend: The regression line shows a positive slope, suggesting higher study hours correlate with increased marks.
- Data Spread: Points cluster along the trend line, though with moderate dispersion, indicating:
  - Consistency in the relationship
  - Presence of other influencing factors



## 11. CORRELATION ANALYSIS

### 11.1. Question 11 - Calculate the coefficient of correlation between study hours and marks scored.

#### Key Finding:

The Pearson correlation coefficient between weekly study hours and academic marks is:

**$r = 0.084$**

#### Interpretation:

1. Strength of Relationship
  - The near-zero value (0.084) indicates an extremely weak positive correlation
  - Practical interpretation: No meaningful linear relationship exists
2. Statistical Significance
  - While not shown in the output, typically:
    - If  $p\text{-value} > 0.05$ : Not statistically significant
    - If  $p\text{-value} \leq 0.05$ : Significant (unlikely due to small  $r$ -value)

```
# --- Question 11: Coefficient of Correlation ---

# Calculate Pearson correlation coefficient
corr_coeff, p_value = pearsonr(    "coeff": Unknown word.
    sample['Study Hours (per week)'],
    sample['Marks (out of 100)']
)

# Display results
print("\033[1m=== Pearson Correlation Coefficient ===\033[0m")
print(f"Correlation Coefficient (r): {corr_coeff:.3f}")    "coeff"

[15]

...    === Pearson Correlation Coefficient ===
      Correlation Coefficient (r): 0.084
```

## **12. REGRESSION ANALYSIS & PREDICTION**

### **12.1. Question 12 - Find the regression lines of: a. Marks Scored on Study Hours, b. Study Hours on Marks Scored.**

#### **Objective:**

This analysis establishes predictive relationships between study hours and academic marks by deriving two regression lines:

1. Marks Scored as a function of Study Hours
2. Study Hours as a function of Marks Scored

The goal is to quantify how changes in one variable predict the other.

#### **Regression Equations:**

1. Predicting Marks from Study Hours:

$$\text{Marks} = 73.84 + 0.25 \times \text{Study\_Hours}$$

Interpretation:

- Intercept (73.84): Expected marks for zero study hours (theoretical, not practical).
- Slope (0.25): Each additional hour of weekly study predicts a 0.25-point increase in marks.

2. Predicting Study Hours from Marks:

$$\text{Study\_Hours} = 11.19 + 0.03 \times \text{Marks}$$

Interpretation:

- Intercept (11.19): Base study hours even for minimal marks.
- Slope (0.03): Each 1-point mark increase predicts 0.03 more weekly study hours.

```

# --- Question 12: Regression Lines ---
from scipy.stats import pearsonr      Import "scipy.stats" could not be resolved

# Extract data from the predefined sample
study_hours = sample['Study Hours (per week)']
marks = sample['Marks (out of 100)']

# Calculate required statistics
study_hours_mean = study_hours.mean()
study_hours_std = study_hours.std(ddof=1) # Sample standard deviation "ddof
marks_mean = marks.mean()
marks_std = marks.std(ddof=1)          "ddof": Unknown word.
corr_coeff, _ = pearsonr(study_hours, marks) "coeff": Unknown word.

# Regression line 1: Marks = a + b * Study_Hours
b_marks_on_hours = corr_coeff * (marks_std / study_hours_std) "coeff": Unkn
a_marks_on_hours = marks_mean - b_marks_on_hours * study_hours_mean
equation1 = f"Marks = {a_marks_on_hours:.2f} + {b_marks_on_hours:.2f} *
Study_Hours"

# Regression line 2: Study_Hours = a' + b' * Marks
b_hours_on_marks = corr_coeff * (study_hours_std / marks_std) "coeff": Unkn
a_hours_on_marks = study_hours_mean - b_hours_on_marks * marks_mean
equation2 = f"Study_Hours = {a_hours_on_marks:.2f} + {b_hours_on_marks:.2f} *
Marks"

# Display results
print("\033[1m=== Regression Lines ===\033[0m")
print(f"a. Marks Scored on Study Hours: {equation1}")
print(f"b. Study Hours on Marks Scored: {equation2}")

```

[16]

```

... === Regression Lines ===
a. Marks Scored on Study Hours: Marks = 73.84 + 0.25 * Study_Hours
b. Study Hours on Marks Scored: Study_Hours = 11.19 + 0.03 * Marks

```

## 12.2. Question 13 - Use the regression line to predict the marks scored by a student who studied for a given number of hours.

### Objective:

In this analysis, the developed regression model in Question 12 is utilized to forecast academic achievement (marks out of 100) from a student's study hours per week. Practical application of the regression equation for planning purposes is the objective here.

### Regression Model Used:

**Marks = 73.84+0.25×Study\_Hours**

**Intercept (a): 73.84**

**Slope (b):** 0.25

**Prediction Example:**

**Input:**

- Study Hours = 12 hours/week

**Calculation:**

**Predicted Marks =  $73.84 + (0.25 \times 12) = 76.80$**

```
# --- Question 13: Predict Marks Using Regression Line ---
# Use the regression equation: Marks = a + b * Study_Hours
# Coefficients from Question 12
hours_studied = 12 # Example input (replace with desired value)
predicted_marks = a_marks_on_hours + b_marks_on_hours * hours_studied

# Display results
print("\033[1m=== Prediction Using Regression Line ===\033[0m")
print(f"For {hours_studied} study hours:")
print(f"Predicted Marks = {predicted_marks:.2f}/100")

[17]

...  === Prediction Using Regression Line ===
      For 12 study hours:
      Predicted Marks = 76.80/100
```

## 13. TEST THE SIGNIFICANCE OF THE CORRELATION COEFFICIENT

### 13.1. Question 14 - Test the significance of the correlation coefficient.

#### Objective:

To evaluate whether a statistically significant linear relationship exists between the number of study hours and marks scored by performing a significance test on the Pearson correlation coefficient.

#### Methodology:

We used the Pearson correlation coefficient to examine the linear relationship between Study Hours (per week) and Marks (out of 100) in the sample. The correlation coefficient  $r$  and its associated p-value were calculated using the `pearsonr()` function from `scipy.stats`.

```
▶ ▾  
# --- Question 14: Significance of Correlation Coefficient ---  
# Recalculate correlation and p-value (if variables are not retained from Q11)  
study_hours = sample['Study Hours (per week)']  
marks = sample['Marks (out of 100)']  
corr_coeff, p_value = pearsonr(study_hours, marks)    "coeff": Unknown word.  
  
# Define significance levels  
alpha_5 = 0.05  
alpha_1 = 0.01
```

#### Results:

```
# Display results  
print("\033[1m=== Significance Test for Correlation Coefficient ===\033[0m")  
print(f"Correlation Coefficient (r): {corr_coeff:.3f}")    "coeff": Unknown word.  
print(f"P-value: {p_value:.4f}")  
  
..    === Significance Test for Correlation Coefficient ===  
      Correlation Coefficient (r): 0.084  
      P-value: 0.5982
```

#### Hypothesis Testing:

- **Null Hypothesis ( $H_0$ ):** There is no significant correlation between study hours and marks.
- **Alternative Hypothesis ( $H_1$ ):** There is a significant correlation between study hours

and marks.

#### Decision criteria:

- If  $p\text{-value} < 0.01 \rightarrow$  Reject  $H_0$  at both 1% and 5% significance levels
- If  $0.01 \leq p\text{-value} < 0.05 \rightarrow$  Reject  $H_0$  at 5% level only
- If  $p\text{-value} \geq 0.05 \rightarrow$  Fail to reject  $H_0$

```
# Hypothesis testing at 5% and 1% significance levels
print("\n\033[1mConclusion:\033[0m")
if p_value < alpha_1:
    print("Reject H0 at both 1% and 5% levels. Correlation is statistically significant.")
elif p_value < alpha_5:
    print("Reject H0 at 5% level but not at 1%. Correlation is significant at 5%.")
else:
    print("Fail to reject H0 at both levels. No significant correlation exists.")
```

#### Conclusion:

Fail to reject  $H_0$  at both levels. No significant correlation exists.

### 13.2. Question 15 - Test the significance of the regression coefficient.

#### Objective:

To check if the regression coefficient (slope) in the linear relationship between study hours and marks obtained is statistically significant. This test indicates whether study hours significantly predict the marks

#### Methodology:

With the use of the correlation coefficient  $rr$  from the previous analysis (Question 14), the  $t$ -statistic to test for significance of the regression coefficient  $bb$  was computed. Mathematically, the test is the same as testing the correlation coefficient but is interpreted within linear regression.

#### Hypothesis Testing:

- **Null Hypothesis ( $H_0$ ):** The regression coefficient  $b=0$  (no linear relationship)
- **Alternative Hypothesis ( $H_1$ ):** The regression coefficient  $b \neq 0$

#### Decision thresholds:

- If  $p\text{-value} < 0.01 \rightarrow$  Reject  $H_0$  at both 1% and 5% levels
- If  $0.01 \leq p\text{-value} < 0.05 \rightarrow$  Reject  $H_0$  at 5% level only
- If  $p\text{-value} \geq 0.05 \rightarrow$  Fail to reject  $H_0$



```

# --- Question 15: Significance of Regression Coefficient (Corrected) ---
import numpy as np      Import "numpy" could not be resolved
from scipy.stats import t      Import "scipy.stats" could not be resolved

# Regression coefficient (b) and correlation coefficient (r) from Q11-Q12
b = b_marks_on_hours
r = corr_coeff # From Q11 (e.g., r = 0.150)      "coeff": Unknown word.
n = len(sample) # Sample size (n = 42)

# Correct t-statistic (equivalent to correlation test)
t_stat = r * np.sqrt(n - 2) / np.sqrt(1 - r**2)
df = n - 2
p_value = 2 * (1 - t.cdf(abs(t_stat), df)) # Two-tailed test

# Display results
print("\n=== Significance Test for Regression Coefficient ===\n")
print(f"Regression Coefficient (b): {b:.3f}")
print(f"t-statistic: {t_stat:.3f}")
print(f"P-value: {p_value:.4f}")

# Hypothesis testing (same conclusion as Q14)
alpha_5 = 0.05
alpha_1 = 0.01

print("\nConclusion:\n")
if p_value < alpha_1:
    print("Reject H₀ at both levels.")
elif p_value < alpha_5:
    print("Reject H₀ at 5% only.")
else:
    print("Fail to reject H₀ at both levels.")

```

[19]

```

...  === Significance Test for Regression Coefficient ===
      Regression Coefficient (b): 0.247
      t-statistic: 0.531
      P-value: 0.5982

      Conclusion:
      Fail to reject H₀ at both levels.

```

## **14. REFERENCES**