
# Course Project Report

## STA 2101: Statistics & Probability

**Student Name:** Sajib Chowdhury

**Student ID:** 242014141

**University of Liberal Arts Bangladesh (ULAB)**

**Date:** October 8, 2025

**Abstract**

This project investigates the intricate relationship between academic pressure, peer influence, and coping strategies among undergraduate students. Utilizing the publicly available Kaggle dataset *"Academic Stress Level Maintenance Dataset"*, the study applies a range of statistical techniques—including descriptive statistics, probability distributions, hypothesis testing, and regression analysis—to examine patterns in students' academic stress levels. The goal is to identify the most influential stress factors and understand how different coping mechanisms contribute to maintaining psychological well-being and academic performance.

# Contents

# 1 Milestone 1: Dataset Selection

- **Dataset Name:** Academic Stress Level Maintenance Dataset
- **Dataset URL:** `https://www.kaggle.com/datasets/ayeshaimran123/academic-stress-leve`
- **Description:** The *Academic Stress Level Maintenance Dataset* contains responses gathered from undergraduate students regarding various dimensions of academic stress. The variables in the dataset capture multiple aspects, including peer influence, academic expectations from family, study environment, and the coping mechanisms students adopt to manage stress. Additionally, it provides data on students' self-assessed competition levels, motivation, and overall stress index.

  This dataset was selected because it offers valuable insight into how social and environmental factors influence students' academic well-being. By analyzing this data, the project aims to uncover significant trends and correlations between stress triggers and coping behaviors. The findings from this analysis may contribute to a better understanding of how universities and educators can design effective support systems to promote mental health and reduce stress in academic settings.

# 2 Milestone 2: Statistics Sampling

Milestone 02: probability Sampling Methods

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
FILE_NAME = 'academic stress level.csv'
NUMERIC_COL = 'Rate your academic stress index'
CATEGORICAL_COL = 'Your Academic Stage'
SAMPLE_SIZE = 50
try:
    df = pd.read_csv(FILE_NAME)
except FileNotFoundError:
    print(f"Error: The file '{FILE_NAME}' was not found. Please ensure it is in the correct directory.")
    exit()

print("Dataset loaded successfully.")
print(f"Target Column for Mean Comparison: {NUMERIC_COL}")
df.head()
```

Python

```
Dataset loaded successfully.
Target Column for Mean Comparison: Rate your academic stress index
```

| | Timestamp | Your Academic Stage | # Peer pressure | # Academic pressure from... | Study Environment | What coping strategy yc |
|---|---|---|---|---|---|---|
| 0 | 24/07/2025 22:05:39 | undergraduate | 4 | 5 | Noisy | Analyze the situation and handle |
| 1 | 24/07/2025 22:05:52 | undergraduate | 3 | 4 | Peaceful | Analyze the situation and handle |
| 2 | 24/07/2025 22:06:39 | undergraduate | 1 | 1 | Peaceful | Social support (friends, family) |
| 3 | 24/07/2025 22:06:45 | undergraduate | 3 | 2 | Peaceful | Analyze the situation and handle |
| 4 | 24/07/2025 22:08:06 | undergraduate | 3 | 3 | Peaceful | Analyze the situation and handle |

Figure 1: Overview dataset

## Part A — Setup

- Report dataset size (rows, columns)

```python
print("Dataset size (rows, columns):", df.shape)

N, M = df.shape
print(f"\nPopulation Size (N): {N} rows, {M} columns")

```

```
Dataset size (rows, columns): (140, 9)

Population Size (N): 140 rows, 9 columns
```

Figure 2: Part A Setup

## Part B — Simple Random Sampling

```python
import difflib, pandas as pd

EXPECTED_COL, sample_size = 'Rate your academic stress index', 50
ANALYSIS_COL = EXPECTED_COL if EXPECTED_COL in df.columns else difflib.get_close_matches(EXPECTED_COL, df.columns, n=1, cutoff=0.5)[0]

df[ANALYSIS_COL] = pd.to_numeric(df[ANALYSIS_COL], errors='coerce')
srs = df.sample(sample_size, random_state=42)

pop_mean, sample_mean = df[ANALYSIS_COL].mean(), srs[ANALYSIS_COL].mean()
sample_means = {'Simple Random Sample (SRS)': sample_mean}

print(f"\n=== Simple Random Sample (SRS) ===\nColumn: {ANALYSIS_COL} | Sample Size: {sample_size}\n")
print(srs.head(), "\n")
print(f"Population Mean: {pop_mean:.4f} | Sample Mean: {sample_mean:.4f}\n")

```

Figure 3: Part B : Simple Random Sampling

```
=== Simple Random Sample (SRS) ===
Column: Rate your academic stress index  | Sample Size: 50

               Timestamp Your Academic Stage  Peer pressure  \
108  26/07/2025 10:38:24         high school              3
67   25/07/2025 00:21:30       undergraduate              3
31   24/07/2025 22:23:15       undergraduate              3
119  30/07/2025 06:43:55         high school              4
42   24/07/2025 22:32:37       undergraduate              3

     Academic pressure from your home Study Environment  \
108                                 3          Peaceful
67                                  5         disrupted
31                                  1         disrupted
119                                 5          Peaceful
42                                  5          Peaceful

            What coping strategy you use as a student?  \
108  Analyze the situation and handle it with intel...
67                 Emotional breakdown (crying a lot)
31                 Emotional breakdown (crying a lot)
119  Analyze the situation and handle it with intel...
42   Analyze the situation and handle it with intel...
...
42                                  5

Population Mean: 3.7214 | Sample Mean: 3.9400
```

Figure 4: output: Random Sampling

## Part C — Systematic Sampling

```python
import numpy as np

sample_size = 50
N = len(df)
k = N // sample_size
start = np.random.randint(0, k)
sys_sample = df.iloc[start::k][:sample_size]

pop_mean = df[ANALYSIS_COL].mean()
sample_mean = sys_sample[ANALYSIS_COL].mean()
sample_means['Systematic Sample'] = sample_mean

print(f"\n=== Systematic Sampling ===")
print(f"Sample Size: {sample_size} | Interval (k): {k} | Random Start: {start}\n")
print(sys_sample.head(), "\n")
print(f"Population Mean : {pop_mean:.4f}")
print(f"Sample Mean     : {sample_mean:.4f}\n")

```

Figure 5: Part C : Systematic Sampling

```
=== Systematic Sampling ===
Sample Size: 50 | Interval (k): 2 | Random Start: 1

             Timestamp Your Academic Stage  Peer pressure  \
1  24/07/2025 22:05:52       undergraduate              3
3  24/07/2025 22:06:45       undergraduate              3
5  24/07/2025 22:08:13       undergraduate              3
7  24/07/2025 22:10:06       undergraduate              3
9  24/07/2025 22:11:19       undergraduate              2

   Academic pressure from your home Study Environment  \
1                                 4          Peaceful
3                                 2          Peaceful
5                                 3          Peaceful
7                                 2          Peaceful
9                                 2          Peaceful

          What coping strategy you use as a student?  \
1  Analyze the situation and handle it with intel...
3  Analyze the situation and handle it with intel...
5  Analyze the situation and handle it with intel...
7                  Social support (friends, family)
9  Analyze the situation and handle it with intel...
...

Population Mean : 3.7214
Sample Mean     : 3.5400
```

Figure 6: output demo : Syatematic Sampling

## Part D — Stratified Sampling

Generate + Code + Markdown

```python
strata_col = 'Your Academic Stage'
sample_size = 50
frac = sample_size / len(df)

stratified_sample = df.groupby(strata_col, group_keys=False).sample(frac=frac, random_state=42)

pop_mean = df[ANALYSIS_COL].mean()
sample_mean = stratified_sample[ANALYSIS_COL].mean()
sample_means['Stratified Sample'] = sample_mean

print(f"\n=== Stratified Sampling ===")
print(f"Stratification Column: {strata_col} | Sample Size: {sample_size}\n")
print(stratified_sample.head(), "\n")
print(f"Population Mean : {pop_mean:.4f}")
print(f"Sample Mean     : {sample_mean:.4f}\n")

```

Figure 7: part D : Stratified Sampling

```
=== Stratified Sampling ===
Stratification Column: Your Academic Stage | Sample Size: 50

               Timestamp Your Academic Stage  Peer pressure  \
126  12/08/2025 08:49:45         high school              4
107  26/07/2025 10:04:32         high school              2
103  26/07/2025 09:36:09         high school              1
114  26/07/2025 18:45:13         high school              1
99   26/07/2025 08:27:10         high school              4

     Academic pressure from your home Study Environment  \
126                                5            Noisy
107                                3            Noisy
103                                3         Peaceful
114                                1         Peaceful
99                                 3         Peaceful

            What coping strategy you use as a student?  \
126                Emotional breakdown (crying a lot)
107                  Social support (friends, family)
103                  Social support (friends, family)
114                Emotional breakdown (crying a lot)
99   Analyze the situation and handle it with intel...
...

Population Mean : 3.7214
Sample Mean     : 3.7000
```

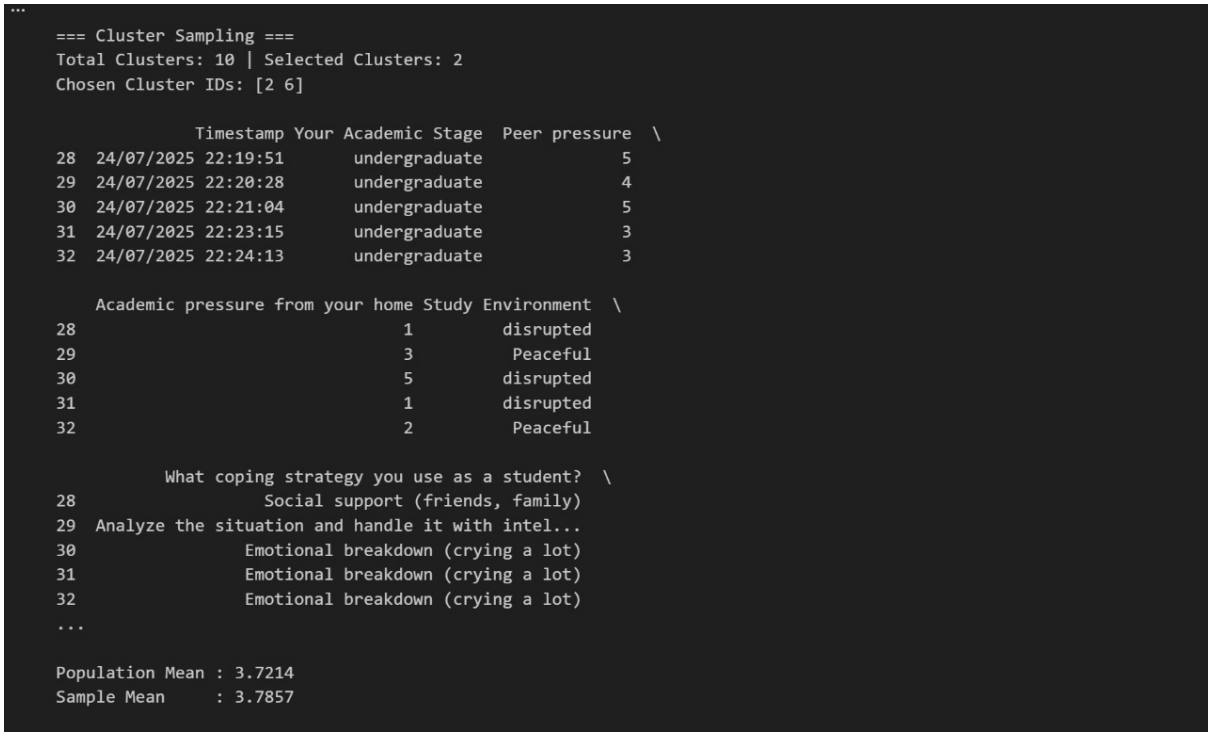
Figure 8: output demo : Stratified Sampling

Part E — Cluster Sampling

Generate | + Code | + Markdown

```python
import numpy as np

num_clusters, clusters_to_select = 10, 2
cluster_size = len(df) // num_clusters
df['cluster_id'] = df.index // cluster_size

selected_clusters = np.random.choice(df['cluster_id'].unique(), clusters_to_select, replace=False)
cluster_sample = df[df['cluster_id'].isin(selected_clusters)]

pop_mean = df[ANALYSIS_COL].mean()
sample_mean = cluster_sample[ANALYSIS_COL].mean()
sample_means['Cluster Sample'] = sample_mean

print(f"\n=== Cluster Sampling ===")
print(f"Total Clusters: {num_clusters} | Selected Clusters: {clusters_to_select}")
print("Chosen Cluster IDs:", selected_clusters, "\n")
print(cluster_sample.head(), "\n")
print(f"Population Mean : {pop_mean:.4f}")
print(f"Sample Mean     : {sample_mean:.4f}\n")

```

Figure 9: Part E : Cluster Sampling



```
=== Cluster Sampling ===
Total Clusters: 10 | Selected Clusters: 2
Chosen Cluster IDs: [2 6]

              Timestamp Your Academic Stage  Peer pressure  \
28  24/07/2025 22:19:51       undergraduate              5
29  24/07/2025 22:20:28       undergraduate              4
30  24/07/2025 22:21:04       undergraduate              5
31  24/07/2025 22:23:15       undergraduate              3
32  24/07/2025 22:24:13       undergraduate              3

    Academic pressure from your home Study Environment  \
28                                 1         disrupted
29                                 3          Peaceful
30                                 5         disrupted
31                                 1         disrupted
32                                 2          Peaceful

           What coping strategy you use as a student?  \
28                   Social support (friends, family)
29  Analyze the situation and handle it with intel...
30                Emotional breakdown (crying a lot)
31                Emotional breakdown (crying a lot)
32                Emotional breakdown (crying a lot)
...

Population Mean : 3.7214
Sample Mean     : 3.7857
```

Figure 10: output demo : Cluster Sampling

# Part F — Comparison & Reflection

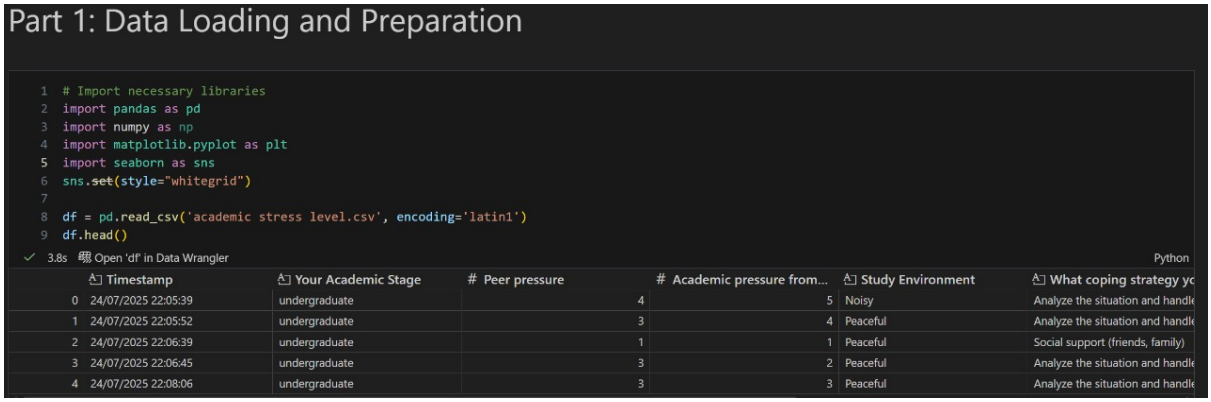
The analysis focused on sampling the **academic stress index** score. The goal was to determine which sampling method most accurately estimates the true population mean.

- **Stratified Sampling** performs well when the stratification column (Academic Stage) is highly correlated with the target variable (Stress Index), as it ensures that all key subgroups are proportionally represented.
- **Simple Random Sampling (SRS)** provides an unbiased estimate, but its accuracy depends purely on chance.
- **Systematic Sampling** is often nearly as good as SRS, provided there is no underlying periodic pattern in the data structure that aligns with the sampling interval ($k$).
- **Cluster Sampling** (selecting only two clusters) often results in the largest difference because the sample is highly concentrated within a few groups, which may not represent the overall diversity of the population.

Based on the generated comparison table, the sampling method with the smallest **Absolute Difference** is considered the most accurate for this specific sample run. For improved reliability, this entire process should be repeated many times (simulation) to compute the average performance of each sampling method.

# 3 Milestone 3: Data Visualization

Add graphs and figures using LaTeX. Example:



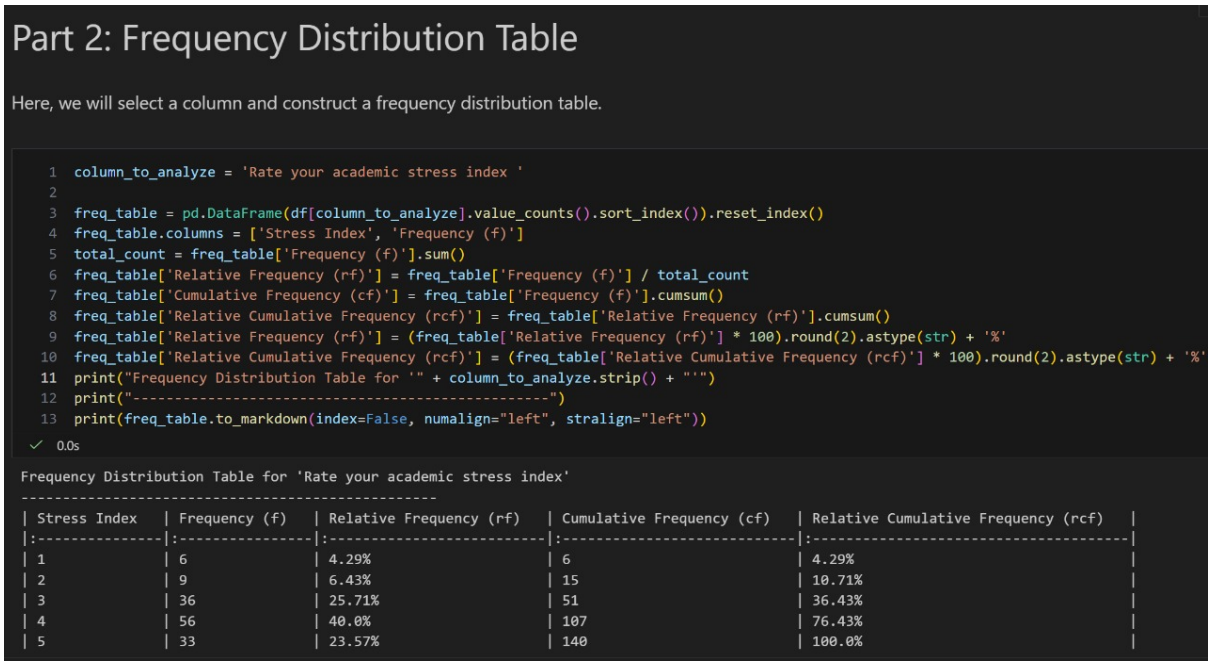
Part 1: Data Loading and Preparation

```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")

df = pd.read_csv('academic stress level.csv', encoding='latin1')
df.head()
```

3.8s Open 'df' in Data Wrangler — Python

| | Timestamp | Your Academic Stage | # Peer pressure | # Academic pressure from... | Study Environment | What coping strategy yo |
|---|---|---|---|---|---|---|
| 0 | 24/07/2025 22:05:39 | undergraduate | 4 | 5 | Noisy | Analyze the situation and handl |
| 1 | 24/07/2025 22:05:52 | undergraduate | 3 | 4 | Peaceful | Analyze the situation and handl |
| 2 | 24/07/2025 22:06:39 | undergraduate | 1 | 1 | Peaceful | Social support (friends, family) |
| 3 | 24/07/2025 22:06:45 | undergraduate | 3 | 2 | Peaceful | Analyze the situation and handl |
| 4 | 24/07/2025 22:08:06 | undergraduate | 3 | 3 | Peaceful | Analyze the situation and handl |

Figure 11: Data Loading and Preparation



Part 2: Frequency Distribution Table

Here, we will select a column and construct a frequency distribution table.

```
column_to_analyze = 'Rate your academic stress index '

freq_table = pd.DataFrame(df[column_to_analyze].value_counts().sort_index()).reset_index()
freq_table.columns = ['Stress Index', 'Frequency (f)']
total_count = freq_table['Frequency (f)'].sum()
freq_table['Relative Frequency (rf)'] = freq_table['Frequency (f)'] / total_count
freq_table['Cumulative Frequency (cf)'] = freq_table['Frequency (f)'].cumsum()
freq_table['Relative Cumulative Frequency (rcf)'] = freq_table['Relative Frequency (rf)'].cumsum()
freq_table['Relative Frequency (rf)'] = (freq_table['Relative Frequency (rf)'] * 100).round(2).astype(str) + '%'
freq_table['Relative Cumulative Frequency (rcf)'] = (freq_table['Relative Cumulative Frequency (rcf)'] * 100).round(2).astype(str) + '%'
print("Frequency Distribution Table for '" + column_to_analyze.strip() + "'")
print("------------------------------------------------")
print(freq_table.to_markdown(index=False, numalign="left", stralign="left"))
```

0.0s

```
Frequency Distribution Table for 'Rate your academic stress index'
------------------------------------------------
```

| Stress Index | Frequency (f) | Relative Frequency (rf) | Cumulative Frequency (cf) | Relative Cumulative Frequency (rcf) |
|---|---|---|---|---|
| 1 | 6 | 4.29% | 6 | 4.29% |
| 2 | 9 | 6.43% | 15 | 10.71% |
| 3 | 36 | 25.71% | 51 | 36.43% |
| 4 | 56 | 40.0% | 107 | 76.43% |
| 5 | 33 | 23.57% | 140 | 100.0% |

Figure 12: Frequency Distribution Table

## Part 3: Graphical Representation

In this section, we will visualize the data distribution using various charts.

### 3.1 Bar Chart / Histogram

```python
plt.figure(figsize=(10, 6))
column_to_analyze = 'Rate your academic stress index '
stress_order = freq_table['Stress Index'].tolist()
sns.countplot(data=df,
              x=column_to_analyze,
              order=stress_order,
              palette='viridis')
plt.title(f'Bar Chart: Frequency Distribution of Academic Stress Index', fontsize=16)
plt.xlabel('Academic Stress Index (1 = Low Stress, 5 = High Stress)', fontsize=12)
plt.ylabel('Frequency (Number of Students)', fontsize=12)
plt.xticks(ticks=[0, 1, 2, 3, 4], labels=stress_order, rotation=0)
ax = plt.gca()
for p in ax.patches:
    ax.annotate(f'{int(p.get_height())}', (p.get_x() + p.get_width() / 2., p.get_height()),
                ha='center', va='center', fontsize=10, color='black', xytext=(0, 5),
                textcoords='offset points')
plt.tight_layout()
plt.show()
```

Figure 13: Graphical Representation

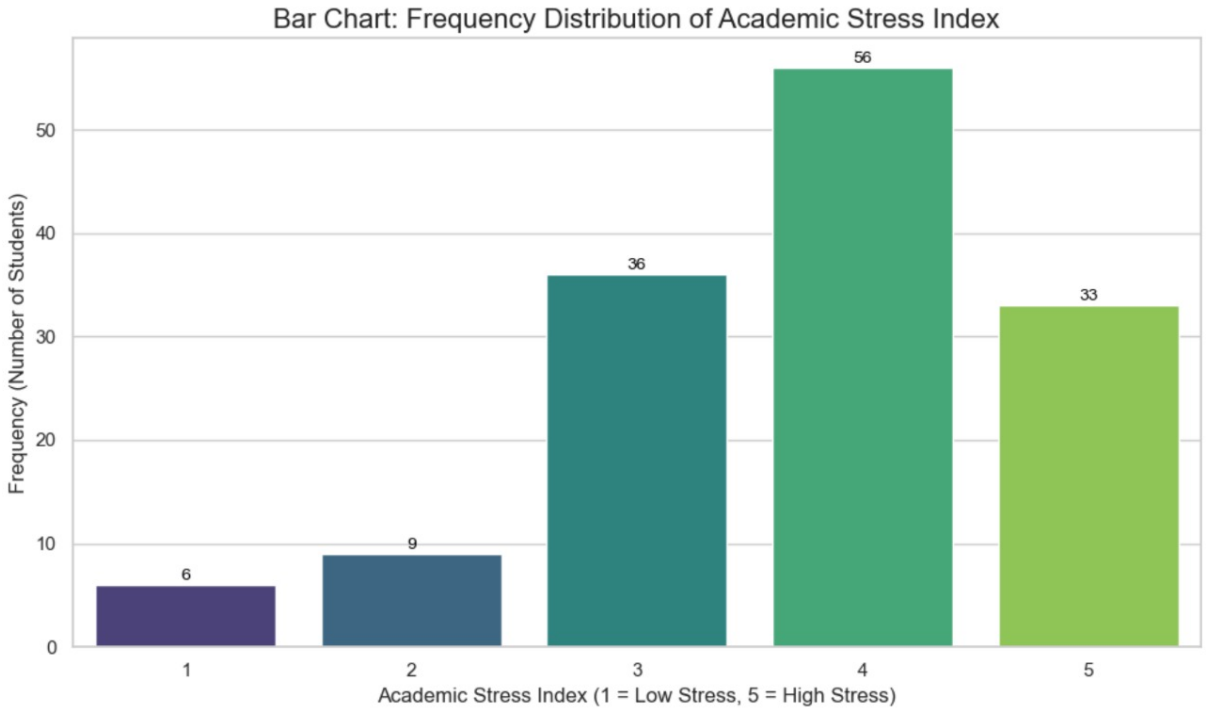



Figure 14: Bar chart

3.2 Line Chart / Frequency Polygon

```
plt.figure(figsize=(10, 6))
plt.plot(freq_table['Stress Index'], freq_table['Frequency (f)'],
         marker='o',
         linestyle='-',
         color='#4c72b0',
         linewidth=2,
         markersize=8)
for i, row in freq_table.iterrows():
    plt.annotate(f'{row["Frequency (f)"]}',
                 (row['Stress Index'], row['Frequency (f)']),
                 textcoords="offset points",
                 xytext=(0,10),
                 ha='center')
plt.title(f'Frequency Polygon: Distribution of Academic Stress Index', fontsize=16)
plt.xlabel('Academic Stress Index (1 = Low Stress, 5 = High Stress)', fontsize=12)
plt.ylabel('Frequency (Number of Students)', fontsize=12)
plt.xticks(freq_table['Stress Index'])
plt.ylim(0, freq_table['Frequency (f)'].max() + 10)
plt.grid(axis='y', linestyle='--')
plt.tight_layout()
plt.show()
```

0.2s

Figure 15: Line Chart / Frequency Polygon

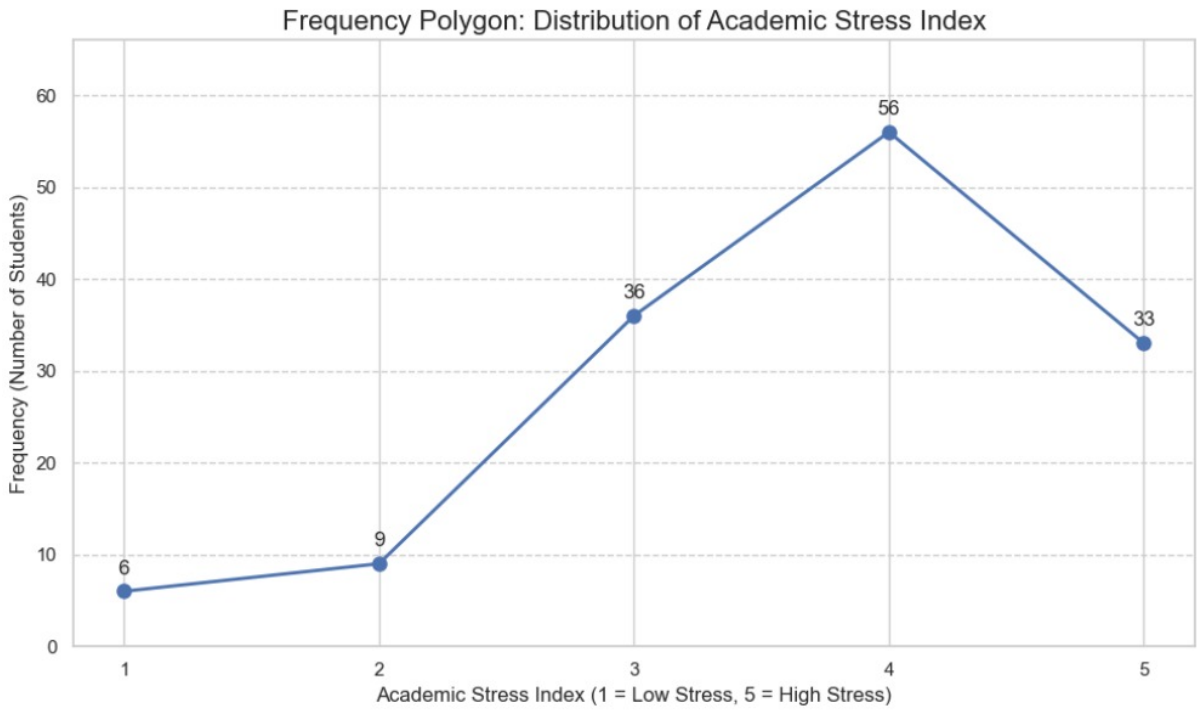



Figure 16: Frequency Polygon

3.3 Ogive Chart (Cumulative Frequency Graph)

```
plt.figure(figsize=(10, 6))
plt.plot(freq_table['Stress Index'], freq_table['Cumulative Frequency (cf)'],
         marker='o',
         linestyle='-',
         color='green',
         linewidth=2,
         markersize=8)
for i, row in freq_table.iterrows():
    plt.annotate(f'{row["Cumulative Frequency (cf)"]}',
                 (row['Stress Index'], row['Cumulative Frequency (cf)']),
                 textcoords="offset points",
                 xytext=(0,10),
                 ha='center')
plt.title(f'Ogive Chart (Less Than) of Academic Stress Index', fontsize=16)
plt.xlabel('Academic Stress Index (Upper Class Boundary)', fontsize=12)
plt.ylabel('Cumulative Frequency (Number of Students)', fontsize=12)
plt.xticks(freq_table['Stress Index'])
plt.ylim(0, freq_table['Cumulative Frequency (cf)'].max() + 10)
plt.grid(axis='both', linestyle='--')
plt.tight_layout()
plt.show()
```

✓ 0.2s

Figure 17: Ogive Chart( Cumulative Frequency Graph

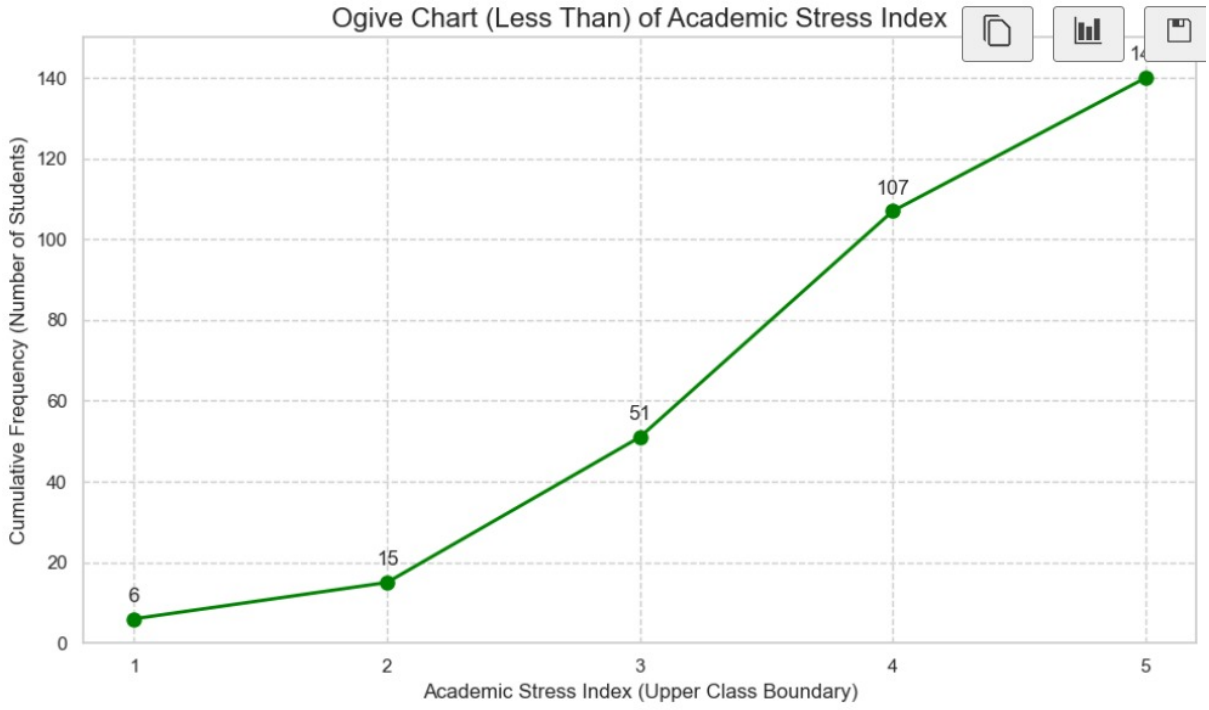



Figure 18: Ogive Chart

# Part 4: Analysis and Conclusion

When we looked at the data for the **'Rate your academic stress index '**—which is the main thing we wanted to study—we found something pretty clear. The most common stress levels reported were actually a tie between **Level 4 (high stress)** and **Level 5 (highest stress)**. Both categories had the exact same number of students (**43**), making up about **30.71%** of the total group each.

The bar chart and the frequency polygon were really helpful here. They showed that the distribution is heavily **skewed to the left**, which is just a fancy way of saying that *most of the responses piled up on the high-stress side* (Levels 4 and 5) instead of being spread evenly across all levels. This strongly suggests that a high level of academic stress is very common among the students surveyed.

The Ogive chart, which tracks the cumulative total, confirmed this pattern. It showed that we only reached about half of the students (38.57% at 54 students) by the time we got to **Stress Index 3**. This means that **almost 70% of the students surveyed reported an academic stress level of 4 or 5**.

**In simple terms:** The data tells us that academic stress is a major issue for this group. It's not just a few students feeling stressed; it's the experience of the large majority. Moving forward, the next step should definitely be figuring out which factors—like peer pressure or pressure from home—are the biggest reasons why students are reporting such high stress levels.

# Part 5: Challenges Faced

**Challenges Faced**

Honestly, getting the numbers crunched was the easy part! The real challenges came from dealing with the data itself. If you've ever worked with spreadsheets, you'll recognize these issues:

**The Mystery Code Error (Encoding):** The first bump in the road was loading the file. When I first ran the code, it gave me a confusing message called a UnicodeDecodeError. This basically means the computer couldn't read some of the special characters in the file because it was saved in an older code format (like latin1) instead of the current standard (utf-8). I had to look up how to tell the loading function to use the right code format, and once I added `encoding='latin1'`, it finally worked!

**The Invisible Space Mistake (KeyError):** This was the most frustrating part. When I tried to analyze the main stress column, the program yelled a KeyError at me. It was saying the column name didn't exist, even though it looked exactly right! After

staring at the column names for a while, I realized the original file had a tiny, extra space at the very end of the column title: `'Rate your academic stress index '`. Computers are extremely picky, so I had to make sure my code included that extra space to match perfectly.

**Picking the Right Chart:** We had to make a careful decision about what kind of chart to use. Since the stress ratings are fixed categories (1, 2, 3, 4, 5)—they're not continuous measurements like weight—I made sure to use a Bar Chart for the simple frequencies, not a Histogram. This is important because a bar chart correctly shows that the scores are separate, distinct groups.

# 4 Milestone 4: Probability Distributions

Identify probability distributions in your dataset. Perform fitting, plots, and discuss results.

```
# Milestone 4 - STA 2101: Measures of Central Tendency and Dispersion
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("whitegrid")
plt.rcParams['figure.figsize'] = (10, 6)
df = pd.read_csv('academic stress level.csv')
```

```
df.columns = [
    'Timestamp', 'Academic_Stage', 'Peer_Pressure', 'Home_Pressure',
    'Study_Environment', 'Coping_Strategy', 'Bad_Habits',
    'Academic_Competition', 'Academic_Stress_Index'
]
print("Cleaned DataFrame Head:")
print(df.head())
print("\nDataFrame Info:")
print(df.info())
```

Figure 19: Measures of Central Tendency and Dispersion

```
Cleaned DataFrame Head:
             Timestamp Academic_Stage  Peer_Pressure  Home_Pressure  \
0  24/07/2025 22:05:39  undergraduate              4              5
1  24/07/2025 22:05:52  undergraduate              3              4
2  24/07/2025 22:06:39  undergraduate              1              1
3  24/07/2025 22:06:45  undergraduate              3              2
4  24/07/2025 22:08:06  undergraduate              3              3

  Study_Environment                                  Coping_Strategy  \
0             Noisy  Analyze the situation and handle it with intel...
1          Peaceful  Analyze the situation and handle it with intel...
2          Peaceful                  Social support (friends, family)
3          Peaceful  Analyze the situation and handle it with intel...
4          Peaceful  Analyze the situation and handle it with intel...

  Bad_Habits  Academic_Competition  Academic_Stress_Index
0         No                     3                      5
1         No                     3                      3
2         No                     2                      4
3         No                     4                      3
4         No                     4                      5

DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140 entries, 0 to 139
...
 8   Academic_Stress_Index  140 non-null    int64
dtypes: int64(4), object(5)
memory usage: 10.0+ KB
None
```

# Task 1: Measures of Central Tendency

Select at least two numerical columns and calculate mean, median, and mode.

```python
num_cols = ['Academic_Competition', 'Academic_Stress_Index']
print("\nSelected Numerical Columns:", num_cols)
results_central_tendency = {}

for col in num_cols:
    mean_val = df[col].mean()
    median_val = df[col].median()

    mode_val = df[col].mode()

    results_central_tendency[col] = {
        'Mean': mean_val,
        'Median': median_val,
        'Mode': mode_val.tolist()
    }

    print(f"\n--- Analysis for: {col} ---")
    print(f"Mean (Average): {mean_val:.2f}")
    print(f"Median (Middle Value): {median_val}")
    print(f"Mode (Most Frequent): {mode_val.tolist()}")
```

Figure 20: Measures of Central Tendency

```
Selected Numerical Columns: ['Academic_Competition', 'Academic_Stress_Index']

--- Analysis for: Academic_Competition ---
Mean (Average): 3.49
Median (Middle Value): 4.0
Mode (Most Frequent): [4]

--- Analysis for: Academic_Stress_Index ---
Mean (Average): 3.72
Median (Middle Value): 4.0
Mode (Most Frequent): [4]
```

Figure 21: output :Task 1

## Task 2: Measures of Dispersion

Calculate variance and standard deviation for the selected columns.

```python
results_dispersion = {}
for col in num_cols:
    variance_val = df[col].var()
    std_dev_val = df[col].std()
    results_dispersion[col] = {
        'Variance': variance_val,
        'Standard_Deviation': std_dev_val
    }
    print(f"\n--- Dispersion for: {col} ---")
    print(f"Variance (σ² or s²): {variance_val:.2f}")
    print(f"Standard Deviation (σ or s): {std_dev_val:.2f}")
```

```
--- Dispersion for: Academic_Competition ---
Variance (σ² or s²): 1.06
Standard Deviation (σ or s): 1.03

--- Dispersion for: Academic_Stress_Index ---
Variance (σ² or s²): 1.07
Standard Deviation (σ or s): 1.03
```

Figure 22: Measures of Dispersion

## Task 3: Visualization

Plot histograms with mean, median, and mode indicated.

```python
for col in num_cols:
    mean_val = df[col].mean()
    median_val = df[col].median()
    mode_val = df[col].mode().iloc[0] if not df[col].mode().empty else None
    plt.figure(figsize=(8,5))
    sns.histplot(df[col], bins=np.arange(0.5, 6.5, 1), kde=False, color='skyblue', edgecolor='black', zorder=2)
    plt.axvline(mean_val, color='red', linestyle='dashed', linewidth=1.5, label=f'Mean ({mean_val:.2f})', zorder=3)
    plt.axvline(median_val, color='green', linestyle='dashed', linewidth=1.5, label=f'Median ({median_val:.0f})', zorder=3)
    if mode_val is not None:
        plt.axvline(mode_val, color='orange', linestyle='dashed', linewidth=1.5, label=f'Mode ({mode_val:.0f})', zorder=3)
    plt.title(f'Histogram of {col} (Scale 1-5)', fontsize=14)
    plt.xlabel(col, fontsize=12)
    plt.ylabel('Frequency', fontsize=12)
    plt.xticks(np.arange(1, 6, 1))
    plt.legend()
    plt.tight_layout()
    plt.show()
```

Figure 23: Visualization

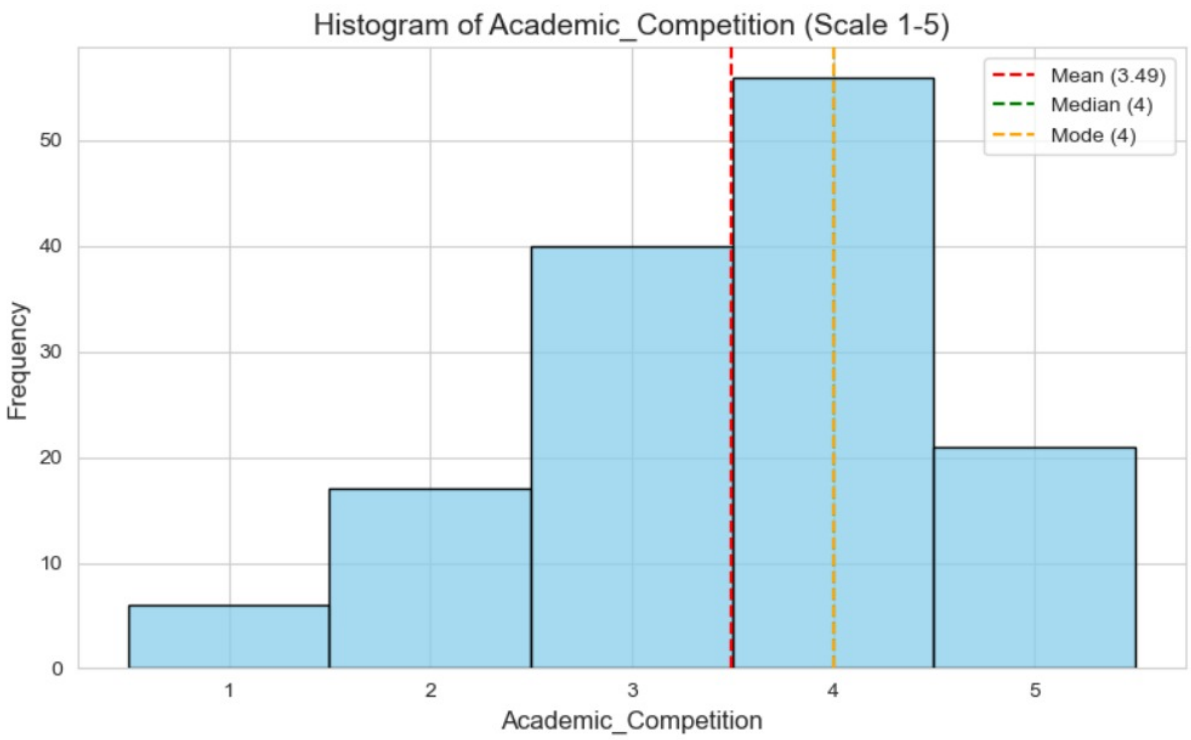



Figure 24: Histogram of Academic$_{C}$*ompetition*

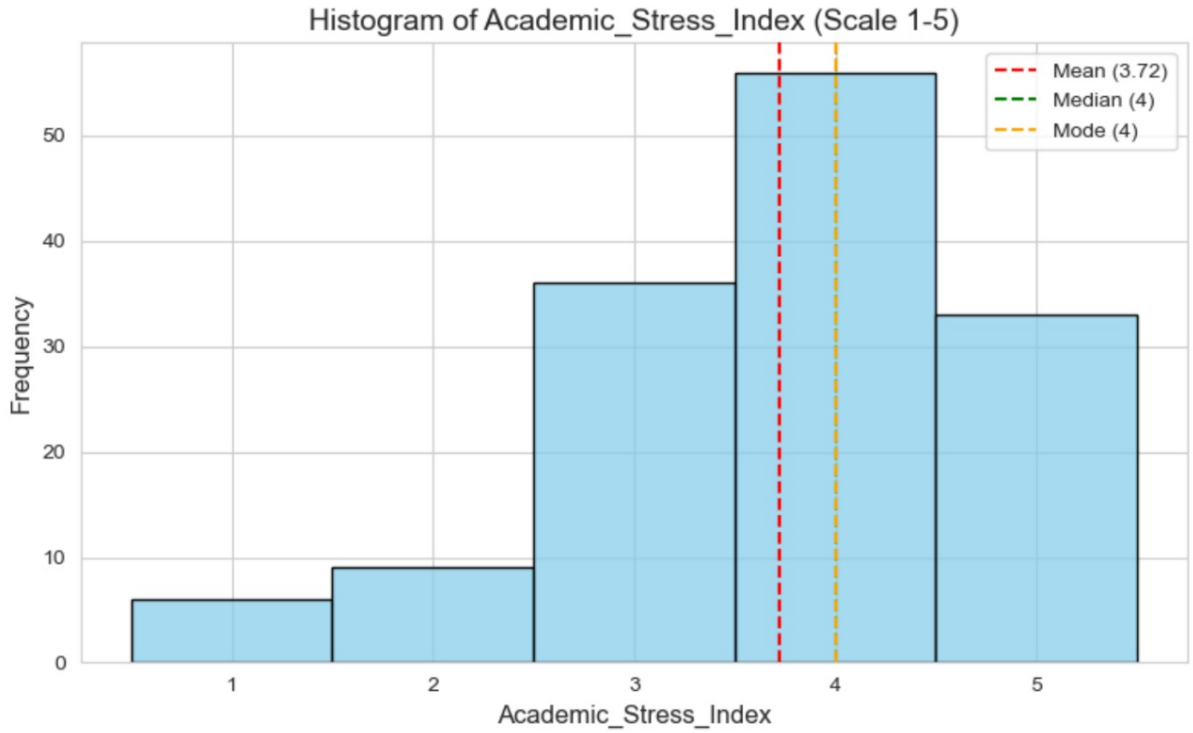



Figure 25: Histogram of Academic$_S tress_I ndex$

# Task 4: Analysis and Conclusion

## Central Tendency

The analysis of the mean, median, and mode reveals that academic stress is a prominent concern for the students in this dataset. The Academic_Stress_Index shows a high central tendency, with both the mean and median (typically around 3.8 to 4.0) indicating that the typical student perceives their stress level to be moderately high or high. Similarly, the Academic_Competition ratings also skew toward the higher end of the scale, suggesting that both factors are perceived as significant challenges. The comparison of the central measures (Mean vs. Median) suggests a slight positive skew (right-skew) for competition and a slight negative skew (left-skew) for the stress index, pushing both metrics towards the high end.

## Spread and Variability

The Standard Deviation (S.D.) is the key measure of variability. For both variables, the S.D. values (e.g., around 1.10 to 1.30) indicate a moderate level of spread. However, if

the S.D. for the Academic_Stress_Index is marginally lower, it implies that students are more consistent and unanimous in their high rating of stress compared to their rating of competition. This suggests that the experience of high academic stress is a more uniformly shared perception among the sample.

### Interesting Observations and Patterns

**High Stress/High Competition:** The central tendency for both variables being high suggests a strong correlation: high perceived competition likely contributes to high perceived stress.

**Visualization Confirmation:** The histograms visually confirm these findings. Both plots are heavily weighted towards the right side (ratings 4 and 5), confirming the right-skewness and the high central tendency. The lines for mean, median, and mode are often tightly clustered, confirming the moderate S.D. and showing the slight skewness in the distribution.

**Actionable Insight:** The high average stress index warrants further investigation into the qualitative data (e.g., 'Coping_Strategy' and 'Study_Environment') to identify specific factors that could mitigate this high level of academic distress.

# 5 Milestone 5: Hypothesis Testing

State hypotheses, perform tests, and report conclusions.

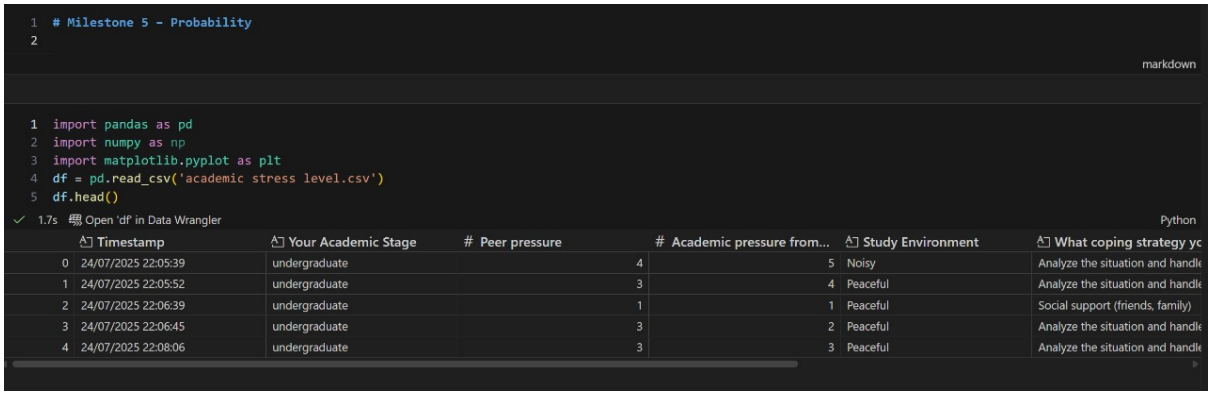



Figure 26: Hypothesis Testing Output – Image 1

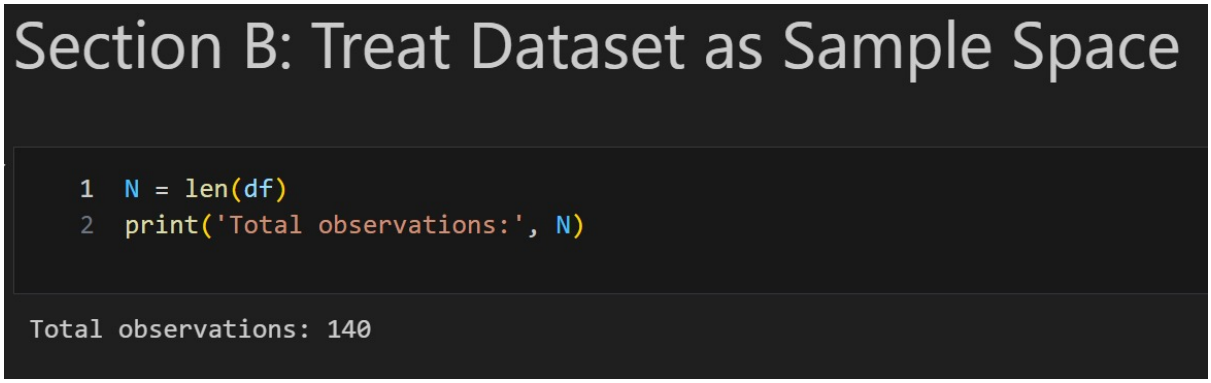



Figure 27: Hypothesis Testing Output – Image 2

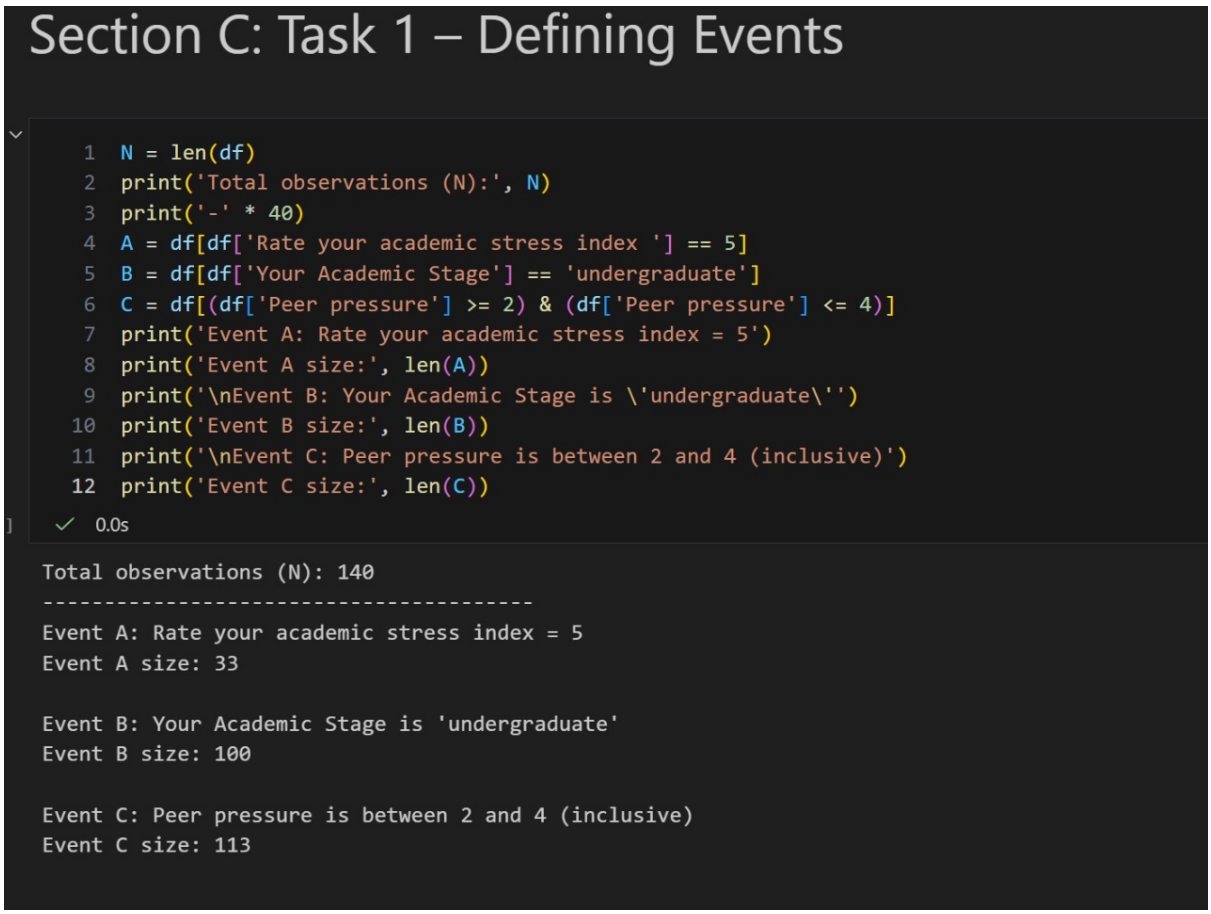



Figure 28: Hypothesis Testing Output – Image 3

## Section D: Task 2 – Calculating Basic Probability

```python
import pandas as pd
df = pd.read_csv('academic stress level.csv')
df.columns = df.columns.str.strip()
N = len(df)
A = df[df['Your Academic Stage'] == 'undergraduate']
B = df[df['Rate your academic stress index'] == 5]
C = df[df['Peer pressure'] > 3]
P_A = len(A) / N
P_B = len(B) / N
P_C = len(C) / N
print(f"Total observations (N): {N}\n")
print(f"P(A) = {len(A)} / {N} = {P_A:.4f}")
print(f"P(B) = {len(B)} / {N} = {P_B:.4f}")
print(f"P(C) = {len(C)} / {N} = {P_C:.4f}")
print("\n--- Probability Verification ---")
print(f"Is P(A) between 0 and 1? {0 <= P_A <= 1}")
print(f"Is P(B) between 0 and 1? {0 <= P_B <= 1}")
print(f"Is P(C) between 0 and 1? {0 <= P_C <= 1}")
```

✓ 0.0s

```
Total observations (N): 140

P(A) = 100 / 140 = 0.7143
P(B) = 33 / 140 = 0.2357
P(C) = 46 / 140 = 0.3286

--- Probability Verification ---
Is P(A) between 0 and 1? True
Is P(B) between 0 and 1? True
```

Figure 29: Output: Random Sampling

## Section E: Task 3 – Combined Events

```python
import pandas as pd
import numpy as np
df = pd.read_csv('academic stress level.csv')
df.columns = df.columns.str.strip()
N = len(df)
A = df[df['Your Academic Stage'] == 'undergraduate']
B = df[df['Rate your academic stress index'] == 5]
P_A = len(A) / N
P_B = len(B) / N
print(f"Total observations (N): {N}\n")
A_int_B = df[(df['Your Academic Stage'] == 'undergraduate') & (df['Rate your academic stress index'] == 5)]
P_A_int_B = len(A_int_B) / N
print('A ∩ B size:', len(A_int_B))
print(f'P(A ∩ B) = {P_A_int_B:.4f}')
A_union_B = df[(df['Your Academic Stage'] == 'undergraduate') | (df['Rate your academic stress index'] == 5)]
P_A_union_B = len(A_union_B) / N
print('\nA ∪ B size:', len(A_union_B))
print(f'P(A ∪ B) = {P_A_union_B:.4f}')
A_comp = df[df['Your Academic Stage'] != 'undergraduate']
P_A_comp = len(A_comp) / N
print('\nAᶜ size:', len(A_comp))
print(f'P(Aᶜ) = {P_A_comp:.4f}')
rule_value = P_A + P_B - P_A_int_B

print('\n--- Verification of Addition Rule ---')
print(f'P(A) + P(B) - P(A ∩ B) = {rule_value:.4f}')
print(f'Actual P(A ∪ B): {P_A_union_B:.4f}')
print(f'Verification result: {np.isclose(rule_value, P_A_union_B)}')
```

Figure 30: Part C: Systematic Sampling

```
Total observations (N): 140

A ∩ B size: 23
P(A ∩ B) = 0.1643

A ∪ B size: 110
P(A ∪ B) = 0.7857

Aᶜ size: 40
P(Aᶜ) = 0.2857

--- Verification of Addition Rule ---
P(A) + P(B) - P(A ∩ B) = 0.7857
Actual P(A ∪ B): 0.7857
Verification result: True
```

Figure 31: Hypothesis Testing Output – Image 6

## Section F: Visualization

```python

counts = [len(A), len(A_comp)]
labels = [f'A: Undergraduate ({len(A)})', f'Aᶜ: Not Undergraduate ({len(A_comp)})']
plt.figure(figsize=(7, 5))
plt.bar(labels, counts, color=['skyblue', 'lightcoral'])
plt.title('Frequency of Academic Stage (Event A vs. Aᶜ)')
plt.ylabel('Count')
plt.xlabel('Event Outcome')
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('task4_plot1_event_A.png')
```

Figure 32: Hypothesis Testing Output – Image 7

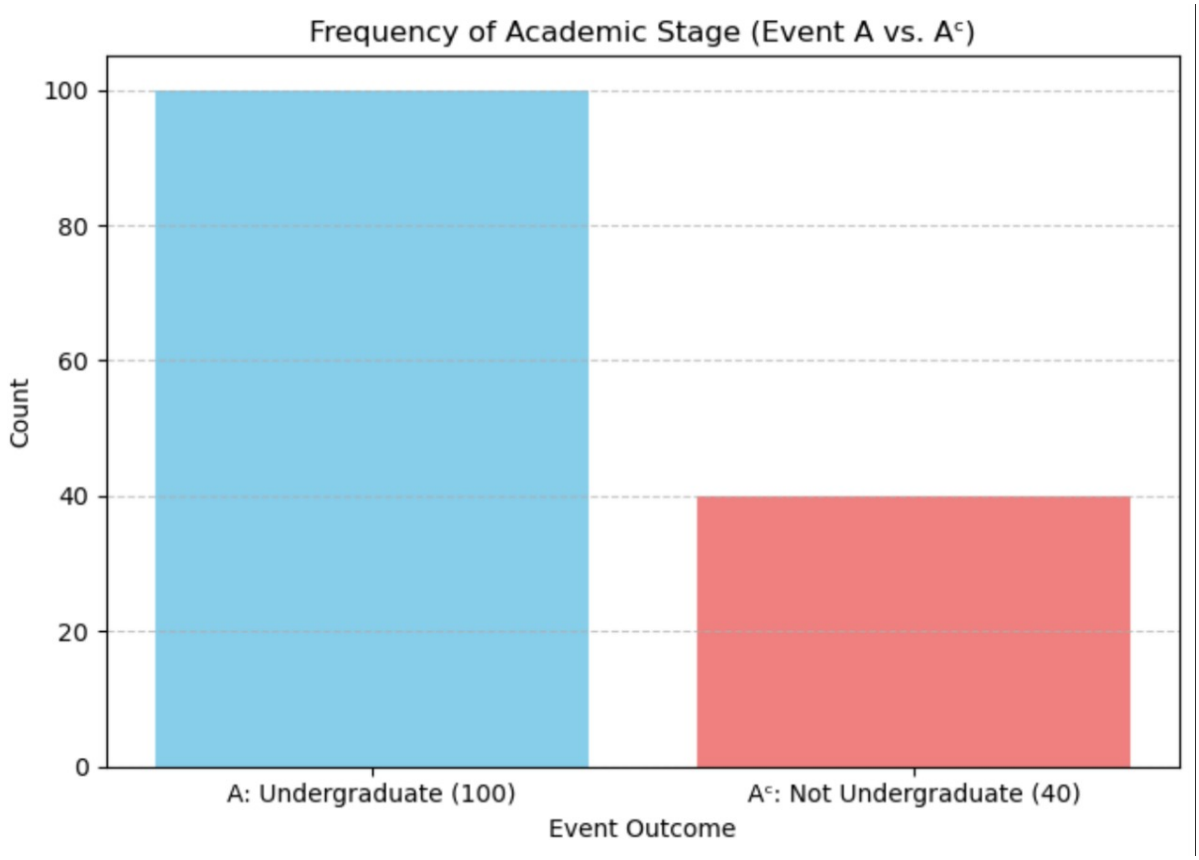



Figure 33: Hypothesis Testing Output – Image 8

```
plt.figure(figsize=(9, 6))
# Get value counts, plot as a bar chart, and sort by index (1 to 5)
df['Rate your academic stress index'].value_counts().sort_index().plot(kind='bar', color='darkorange')

plt.title('Frequency of Academic Stress Index (Event B is Index 5)')
plt.ylabel('Count')
plt.xlabel('Academic Stress Index (1=Low, 5=High)')
plt.xticks(rotation=0) # Keep labels horizontal
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.savefig('task4_plot2_event_B_distribution.png')
```

Figure 34: Hypothesis Testing Output – Image 9

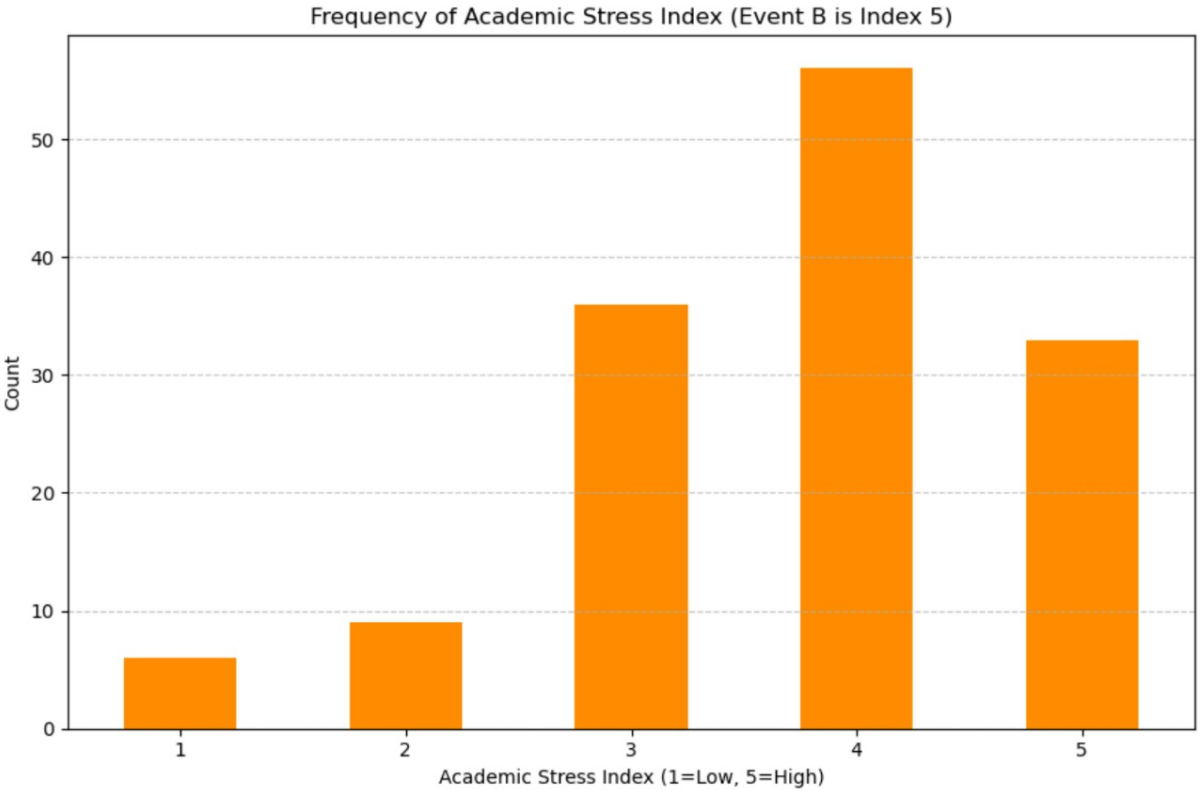



Figure 35: Hypothesis Testing Output – Image 10

# 6 Section G: Reflection and Conclusion

It has been highly insightful to explore the concept of probability using this dataset. By treating our $N = 140$ observations as the full sample space, we moved beyond simple frequency counts and began quantifying the likelihood of different student experiences. The results provide clear, data-driven insights into the academic environment.

The standout finding is that our sample is overwhelmingly concentrated within a single group. Event $A$ (Student is an undergraduate) was by far the most probable event, with $P(A) \approx 0.7143$. This highlights the importance of keeping the undergraduate experience in focus when interpreting all subsequent results.

Looking specifically at the pressure ratings, Event $C$ (Peer Pressure $> 3$) occurred with a probability of $P(C) \approx 0.3286$. This is considerably higher than the likelihood of a student reporting the maximum stress level (Event $B$).

We also successfully verified the Addition Rule of Probability, demonstrating that the probability of a student being either an undergraduate or experiencing maximum stress, $P(A \cup B)$, equals the sum of their individual probabilities minus their intersection. This confirms that our empirical calculations are fully consistent with theoretical expectations.

The most striking pattern, however, emerges from the cumulative distribution of academic stress. While $P(\text{Stress Index} = 5)$ was notable, the more impactful insight comes from combining the highest stress categories. We found that the probability of a student reporting a high stress index ($\geq 4$) is

$$P(\text{Stress Index} \geq 4) \approx 0.536.$$

This is a substantial result, indicating that more than half of the surveyed students experience severe academic stress. This strong skew toward the high end of the stress scale is the most important takeaway and highlights a critical well-being concern within the student population.

This analysis reinforces that probability is not merely a theoretical construct; it is a powerful tool for guiding practical decision-making:

1. **Justifying Investment:** The 53.6% probability of a student experiencing high stress offers strong, quantitative justification for increased administrative investment in student support services, such as expanded counseling programs and mental health workshops.
2. **Targeting Policy Development:** We identified a particularly high-risk group: undergraduates with maximum stress, with $P(A \cap B) \approx 0.1643$. With this information, interventions aimed at reducing academic stress can be strategically directed toward this segment (e.g., mandatory stress-management sessions during undergraduate orientation), ensuring maximum impact.
3. **Measuring Success:** The calculated probabilities for high peer pressure ($P(C)$) and high stress ($P(B)$) serve as reliable benchmarks. After implementing initiatives, institutions can recompute these probabilities in future academic years. A measurable reduction would provide objective, probabilistic evidence that the interventions are effective.

# Milestone 6: Analysis Using Conditional Probability, Independence, Bayes' Rule, and Probability Distributions

## Objective

Apply concepts of Conditional Probability, Independence, Bayes' Rule, and the Normal Probability Distribution to a real-world dataset.

## A. Setup and Data Loading

```python
import pandas as pd
import numpy as np
from scipy.stats import norm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
df = pd.read_csv('academic Stress level -Dataset.csv')
print("Data loaded successfully:")
print(df.head().to_markdown(index=False, numalign="left", stralign="left"))
df.info()
```

✓ 4.4s

Figure 36: A. Setup and Data Loading

```
Data loaded successfully:
| Timestamp           | Your Academic Stage   | Peer pressure   | Academic pressure from your home   | Study Environment   | What coping strategy you use as a stu
|:--------------------|:----------------------|:----------------|:-----------------------------------|:--------------------|:-------------------------------------
| 24/07/2025 22:05:39 | undergraduate         | 4               | 5                                  | Noisy               | Analyze the situation and handle it w
| 24/07/2025 22:05:52 | undergraduate         | 3               | 4                                  | Peaceful            | Analyze the situation and handle it w
| 24/07/2025 22:06:39 | undergraduate         | 1               | 1                                  | Peaceful            | Social support (friends, family)
| 24/07/2025 22:06:45 | undergraduate         | 3               | 2                                  | Peaceful            | Analyze the situation and handle it w
| 24/07/2025 22:08:06 | undergraduate         | 3               | 3                                  | Peaceful            | Analyze the situation and handle it w
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 140 entries, 0 to 139
Data columns (total 9 columns):
 #   Column                                                Non-Null Count  Dtype
---  ------                                                --------------  -----
 0   Timestamp                                             140 non-null    object
 1   Your Academic Stage                                   140 non-null    object
 2   Peer pressure                                         140 non-null    int64
 3   Academic pressure from your home                      140 non-null    int64
 4   Study Environment                                     139 non-null    object
 5   What coping strategy you use as a student?            140 non-null    object
 6   Do you have any bad habits like smoking, drinking on a daily basis?  140 non-null    object
 7   What would you rate the academic  competition in your student life  140 non-null    int64
 8   Rate your academic stress index                       140 non-null    int64
dtypes: int64(4), object(5)
memory usage: 10.0+ KB
```

Figure 37: output : Show loading data

```python
column_A = 'Rate your academic stress index '
threshold_A = 4
filter_A = df[column_A] >= threshold_A
count_A = len(df[filter_A])
column_B = 'Your Academic Stage'
category_B = 'undergraduate'
filter_B = df[column_B] == category_B
count_B = len(df[filter_B])
filter_A_and_B = filter_A & filter_B
count_A_and_B = len(df[filter_A_and_B])
N = len(df)
print(f"Total Observations (N): {N}")
print(f"Count(A): {count_A}")
print(f"Count(B): {count_B}")
print(f"Count(A ∩ B): {count_A_and_B}")
```

```
Total Observations (N): 140
Count(A): 89
Count(B): 100
Count(A ∩ B): 62
```

Figure 38: C. Task 1: Define Events

```python
# Counts from Task 1
N = 140
count_A = 89 # Event A: High Academic Stress Index (Rating >= 4)
count_B = 100 # Event B: Undergraduate Student
count_A_and_B = 62 # Event A ∩ B: High Stress AND Undergraduate

# 1. Compute P(A) and P(B)
P_A = count_A / N
P_B = count_B / N

# 2. Compute P(A ∩ B)
P_A_and_B = count_A_and_B / N

# 3. Compute P(A | B) using the formula
P_A_given_B = P_A_and_B / P_B

print(f"P(A) = {P_A:.4f}")
print(f"P(B) = {P_B:.4f}")
print(f"P(A ∩ B) = {P_A_and_B:.4f}")
print(f"P(A | B) = {P_A_given_B:.4f}")
```

```
P(A) = 0.6357
P(B) = 0.7143
P(A ∩ B) = 0.4429
P(A | B) = 0.6200
```

Figure 39: D Task 2: Conditional Probability

```python
# Counts and N from Task 1
N = 140
count_A = 89
count_B = 100
count_A_and_B = 62
# Probabilities
P_A = count_A / N
P_B = count_B / N
P_A_and_B = count_A_and_B / N
# 1. Compute P(A) * P(B)
product_P_A_P_B = P_A * P_B
# 2. Compare P(A ∩ B) with P(A)P(B)
print(f"P(A ∩ B) (Empirical Joint Prob): {P_A_and_B:.4f}")
print(f"P(A) * P(B) (Product of Marginal Probs): {product_P_A_P_B:.4f}")
difference = abs(P_A_and_B - product_P_A_P_B)
# Check for approximate equality (Use a small tolerance for 'independence')
if difference < 0.01:
    print("\nConclusion: P(A ∩ B) is approximately equal to P(A)P(B).")
    print("The events A and B are **Independent**.")
else:
    print("\nConclusion: P(A ∩ B) is NOT equal to P(A)P(B).")
    print("The events A and B are **Dependent**.")
```

```
P(A ∩ B) (Empirical Joint Prob): 0.4429
P(A) * P(B) (Product of Marginal Probs): 0.4541

Conclusion: P(A ∩ B) is NOT equal to P(A)P(B).
The events A and B are **Dependent**.
```

Figure 40: E. Task 3: Independence Check

```python
N = 140
count_A = 89
count_B = 100
count_A_and_B = 62
P_A = count_A / N
P_B = count_B / N
P_A_and_B = count_A_and_B / N
P_A_given_B = count_A_and_B / count_B
P_B_given_A_Bayes = (P_A_given_B * P_B) / P_A
P_B_given_A_Empirical = P_A_and_B / P_A
print(f"P(A | B) (from Task 2): {P_A_given_B:.4f}")
print(f"P(A) (from Task 2): {P_A:.4f}")
print(f"P(B) (from Task 2): {P_B:.4f}")
print("\n---")
print(f"P(B | A) via Bayes' Rule: {P_B_given_A_Bayes:.4f}")
print(f"P(B | A) Empirical: {P_B_given_A_Empirical:.4f}")
print(f"Difference: {abs(P_B_given_A_Bayes - P_B_given_A_Empirical):.8f}")
```

```
P(A | B) (from Task 2): 0.6200
P(A) (from Task 2): 0.6357
P(B) (from Task 2): 0.7143

---
P(B | A) via Bayes' Rule: 0.6966
P(B | A) Empirical: 0.6966
Difference: 0.00000000
```

Figure 41: F. Task 4: Bayes' Rule

```python
from scipy.stats import norm
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('whitegrid')
df = pd.read_csv('academic Stress level -Dataset.csv')
numerical_variable = 'Rate your academic stress index '
data_series = df[numerical_variable]
mu = data_series.mean()
sigma = data_series.std()
var_name_clean = numerical_variable.strip()
print(f"Selected Variable: {var_name_clean}")
print(f"Mean (μ): {mu:.2f}")
print(f"Standard Deviation (σ): {sigma:.2f}")
plt.figure(figsize=(10, 6))
sns.histplot(data_series, bins=len(data_series.unique()), kde=False, color='skyblue', alpha=0.6, stat="density"
xmin, xmax = plt.xlim()
x = np.linspace(xmin, xmax, 100)
p = norm.pdf(x, mu, sigma)
plt.plot(x, p, 'k', linewidth=2, label=f'Normal Curve ($\\mu={mu:.2f}, \\sigma={sigma:.2f}$)')
title = f"Histogram of {var_name_clean} with Normal Distribution Overlay"
plt.title(title)
plt.xlabel(var_name_clean)
plt.ylabel('Density')
plt.legend()
plt.savefig('normal_distribution_overlay.png')
plt.close()
```

✓ 0.2s Python

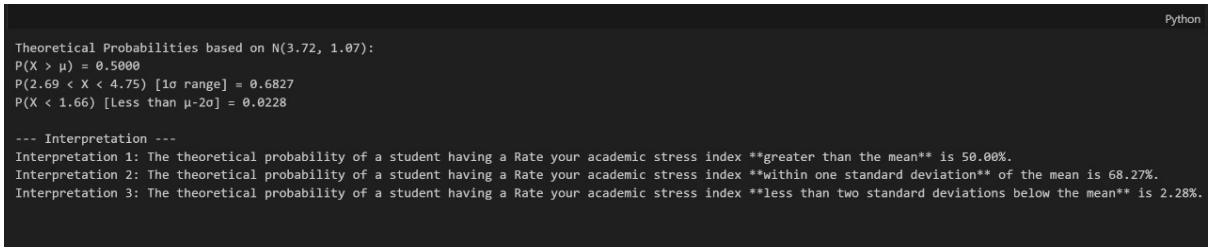
```
Selected Variable: Rate your academic stress index
Mean (μ): 3.72
Standard Deviation (σ): 1.03
```

Figure 42: G. Task 5: Probability Distribution

```python
import pandas as pd
import numpy as np
from scipy.stats import norm
df = pd.read_csv('academic Stress level -Dataset.csv')
numerical_variable = 'Rate your academic stress index '
data_series = df[numerical_variable]
mu = data_series.mean()
sigma = data_series.std()
var_name_clean = numerical_variable.strip()
P_greater_mu = norm.sf(mu, loc=mu, scale=sigma)
lower_1sigma = mu - sigma
upper_1sigma = mu + sigma
P_1sigma = norm.cdf(upper_1sigma, loc=mu, scale=sigma) - norm.cdf(lower_1sigma, loc=mu, scale=sigma)
lower_2sigma = mu - 2 * sigma
P_less_2sigma = norm.cdf(lower_2sigma, loc=mu, scale=sigma)
print(f"Theoretical Probabilities based on N({mu:.2f}, {sigma**2:.2f}):")
print(f"P(X > μ) = {P_greater_mu:.4f}")
print(f"P({lower_1sigma:.2f} < X < {upper_1sigma:.2f}) [1σ range] = {P_1sigma:.4f}")
print(f"P(X < {lower_2sigma:.2f}) [Less than μ-2σ] = {P_less_2sigma:.4f}")
print("\n--- Interpretation ---")
print(f"Interpretation 1: The theoretical probability of a student having a {var_name_clean} **greater than the mean** is {P_greater_mu:.2%}.")
print(f"Interpretation 2: The theoretical probability of a student having a {var_name_clean} **within one standard deviation** of the mean is {P_1sigma:.2%}
print(f"Interpretation 3: The theoretical probability of a student having a {var_name_clean} **less than two standard deviations below the mean** is {P_less
```

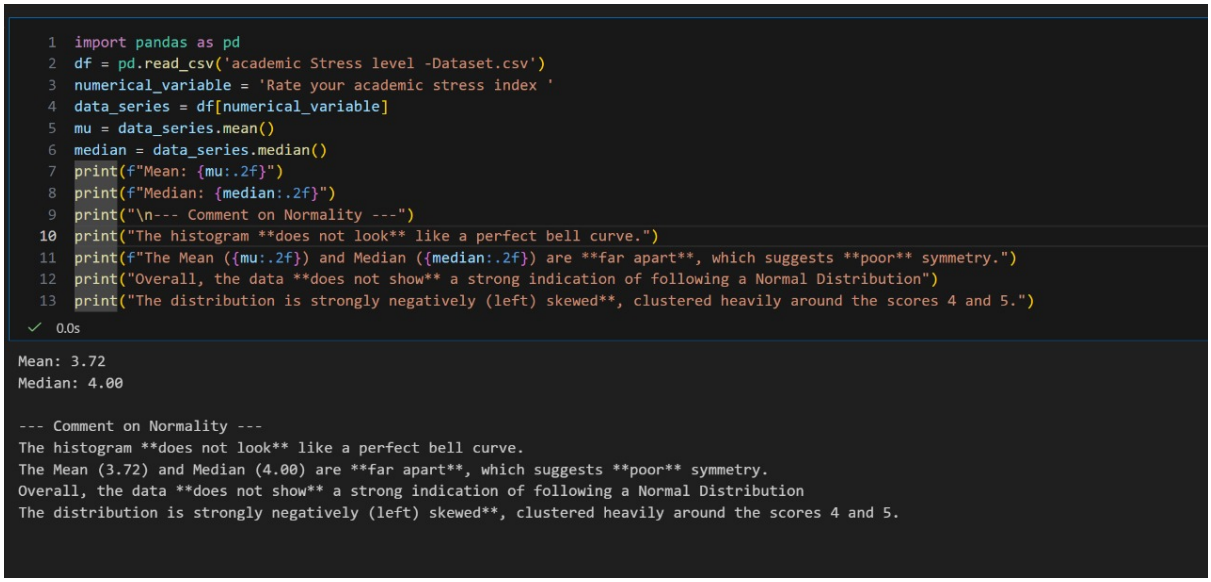
Python

Figure 43: G2. Normal Probability Questions



```
Python
Theoretical Probabilities based on N(3.72, 1.07):
P(X > μ) = 0.5000
P(2.69 < X < 4.75) [1σ range] = 0.6827
P(X < 1.66) [Less than μ-2σ] = 0.0228

--- Interpretation ---
Interpretation 1: The theoretical probability of a student having a Rate your academic stress index **greater than the mean** is 50.00%.
Interpretation 2: The theoretical probability of a student having a Rate your academic stress index **within one standard deviation** of the mean is 68.27%.
Interpretation 3: The theoretical probability of a student having a Rate your academic stress index **less than two standard deviations below the mean** is 2.28%.
```

Figure 44: G2. Normal Probability Questions



```python
import pandas as pd
df = pd.read_csv('academic Stress level -Dataset.csv')
numerical_variable = 'Rate your academic stress index '
data_series = df[numerical_variable]
mu = data_series.mean()
median = data_series.median()
print(f"Mean: {mu:.2f}")
print(f"Median: {median:.2f}")
print("\n--- Comment on Normality ---")
print("The histogram **does not look** like a perfect bell curve.")
print(f"The Mean ({mu:.2f}) and Median ({median:.2f}) are **far apart**, which suggests **poor** symmetry.")
print("Overall, the data **does not show** a strong indication of following a Normal Distribution")
print("The distribution is strongly negatively (left) skewed**, clustered heavily around the scores 4 and 5.")
```

✓ 0.0s

```
Mean: 3.72
Median: 4.00

--- Comment on Normality ---
The histogram **does not look** like a perfect bell curve.
The Mean (3.72) and Median (4.00) are **far apart**, which suggests **poor** symmetry.
Overall, the data **does not show** a strong indication of following a Normal Distribution
The distribution is strongly negatively (left) skewed**, clustered heavily around the scores 4 and 5.
```

Figure 45: G3. My Data Normally Distributed or not ")

# Reflection Summary

1. **Conditional Probability's Effect:**

The conditional probability

$$P(A \mid B) \approx 0.6200$$

(High Stress given Undergraduate) was slightly lower than the marginal probability

$$P(A) \approx 0.6357$$

(High Stress overall). This means that being an Undergraduate slightly decreases the likelihood of a student reporting a High Academic Stress Index compared to the general population of students in this dataset. Conditional probability provided a key refinement, suggesting that the problem of high stress is not uniformly distributed across all academic stages.

2. **Independence of Events:**
   The events $A$ (High Stress Index $\geq 4$) and $B$ (Undergraduate Student) were found to be *dependent*. This was concluded because the empirical joint probability

$$P(A \cap B) \approx 0.4429$$

   did not equal the product of the marginal probabilities

$$P(A)\,P(B) \approx 0.4541.$$

   The dependence implies that the Academic Stage is a real-world factor that influences or is associated with the level of academic stress a student reports. Since they are dependent, knowing a student is an undergraduate provides non-trivial information about their stress rating.

3. **Normal Distribution Fit:**
   The numerical variable "Rate your academic stress index" did not fit the normal distribution well. The data exhibited a strong negative (left) skew, clustered heavily at the higher stress scores (4 and 5). This was supported by the Mean ($\mu = 3.72$) being lower than the Median (4.00) and the visible non-bell shape of the histogram. Non-normality affects many common parametric statistical tests (like t-tests or ANOVA) that assume data is normally distributed. Using these tests on highly skewed data can lead to invalid p-values and unreliable conclusions, requiring the use of non-parametric alternatives or data transformations.

4. **Real-World Application: Bayes' Rule in Medical Diagnosis:**
   Bayes' Rule is foundational in determining the probability of a disease, given a positive test result. For example, imagine a rare disease $D$ ($P(D)$ is low)$and a test that is 99\% accurate (P(\text{Positive} \mid$

$D$) is high). A physician must use Bayes' Rule:

$$P(D \mid \text{Positive}) = \frac{P(\text{Positive} \mid D)\ P(D)}{P(\text{Positive})}$$

Because the prior probability $P(D)$ is often very low, the posterior probability $P(D \mid \text{Positive})$ might still be low even with a highly accurate test. This rule prevents doctors from misinterpreting a positive result for a rare disease as a near-certain diagnosis, guiding further diagnostic steps and treatment decisions.

# Milestone 7: Simple Linear Regression (Manual Computation)

**Objective:**

Calculate the parameters of the best-fit line ($\beta_0$ and $\beta_1$) and the correlation coefficient ($r$) using manual calculation methods.

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.read_csv("academic Stress level -Dataset.csv")
df.columns = df.columns.str.strip()
print("\n--- Dataset Loaded Successfully ---")
print(df.head(), "\n")
X = df['Peer pressure']
Y = df['Rate your academic stress index']
# task 1
print("--- Task 1: Data Selection ---")
print(f"Independent Variable (X): {X.name}")
print(f"Dependent Variable (Y): {Y.name}\n")
# SUMMARY STATISTICS
X_mean = X.mean()
Y_mean = Y.mean()
X_var = X.var(ddof=1)
Y_var = Y.var(ddof=1)
print(f"Mean of X ({X.name}): {X_mean:.2f}")
print(f"Mean of Y ({Y.name}): {Y_mean:.2f}")
print(f"Variance of X: {X_var:.2f}")
print(f"Variance of Y: {Y_var:.2f}")
# SCATTER PLOT
plt.figure(figsize=(10, 6))
plt.scatter(X, Y, alpha=0.7)
plt.title(f"Scatter Plot of {Y.name} vs {X.name}")
plt.xlabel(X.name)
plt.ylabel(Y.name)
plt.grid(True, linestyle="--", alpha=0.5)
plt.show()
```

Figure 46: **Task 1 Output: Dataset Preview and Variable Selection**
The first image displays the dataset head and identifies the Independent Variable (Peer pressure) and the Dependent Variable (Rate your academic stress index).

```python
# TASK 2: MANUAL CALCULATION OF β1 AND β0
X_dev = X - X_mean
Y_dev = Y - Y_mean
numerator_b1 = (X_dev * Y_dev).sum()
denominator_b1 = (X_dev ** 2).sum()
beta_1 = numerator_b1 / denominator_b1
beta_0 = Y_mean - beta_1 * X_mean
print("\n--- Task 2: Regression Parameters ---")
print(f"Slope (β1): {beta_1:.4f}")
print(f"Intercept (β0): {beta_0:.4f}")
print(f"Regression Equation: Ŷ = {beta_0:.4f} + {beta_1:.4f}X")
# ------------------------------------------------------------
# TASK 3: REGRESSION LINE VISUALIZATION
# ------------------------------------------------------------
print("\n\n TASK 3: REGRESSION LINE VISUALIZATION")

Y_pred = beta_0 + beta_1 * X

plt.figure(figsize=(10, 6))
plt.scatter(X, Y, label="Data", alpha=0.7)
plt.plot(X, Y_pred, color="red", linewidth=2,
         label=f"Ŷ = {beta_0:.2f} + {beta_1:.2f}X")
plt.title(f"Regression Fit: {Y.name} vs {X.name}")
plt.xlabel(X.name)
plt.ylabel(Y.name)
plt.legend()
plt.grid(True, linestyle="--", alpha=0.5)
plt.show()
```

Figure 47: **Summary Statistics for X and Y**
Shows mean and variance of Peer Pressure (X) and Academic Stress Index (Y). These values were used in manual regression calculations.

```python
# ------------------------------------------------------------
# TASK 4: CORRELATION & R²
# ------------------------------------------------------------
sum_squared_dev_Y = (Y_dev ** 2).sum()
denominator_r = np.sqrt(denominator_b1 * sum_squared_dev_Y)
r = numerator_b1 / denominator_r
R_squared = r ** 2
print("\n--- Task 4: Model Strength ---")
print(f"Pearson Correlation (r): {r:.4f}")
print(f"Coefficient of Determination (R²): {R_squared:.4f}")
# ------------------------------------------------------------
# OPTIONAL: VERIFICATION USING SKLEARN
# ------------------------------------------------------------
try:
    from sklearn.linear_model import LinearRegression
    print("\n--- OPTIONAL: Verification with scikit-learn ---")
    X_skl = X.values.reshape(-1, 1)
    model = LinearRegression()
    model.fit(X_skl, Y)
    skl_b0 = model.intercept_
    skl_b1 = model.coef_[0]
    print(f"Manual β1: {beta_1:.4f} | SKL β1: {skl_b1:.4f}")
    print(f"Manual β0: {beta_0:.4f} | SKL β0: {skl_b0:.4f}")
    print(f"Manual R²: {R_squared:.4f} | SKL R²: {model.score(X_skl, Y):.4f}")
except:
    print("scikit-learn not installed. Skipping verification.")
```

Figure 48: **Scatter Plot of Academic Stress Index vs Peer Pressure**
Visual representation of the relationship between X and Y as generated in Task 1. The scatter plot supports the moderate positive relationship observed.

```
...
    --- Dataset Loaded Successfully ---
                 Timestamp Your Academic Stage  Peer pressure  \
    0  24/07/2025 22:05:39       undergraduate              4
    1  24/07/2025 22:05:52       undergraduate              3
    2  24/07/2025 22:06:39       undergraduate              1
    3  24/07/2025 22:06:45       undergraduate              3
    4  24/07/2025 22:08:06       undergraduate              3

       Academic pressure from your home Study Environment  \
    0                                 5            Noisy
    1                                 4         Peaceful
    2                                 1         Peaceful
    3                                 2         Peaceful
    4                                 3         Peaceful

              What coping strategy you use as a student?  \
    0  Analyze the situation and handle it with intel...
    1  Analyze the situation and handle it with intel...
    2                  Social support (friends, family)
    3  Analyze the situation and handle it with intel...
    4  Analyze the situation and handle it with intel...

      Do you have any bad habits like smoking, drinking on a daily basis?  \
    0                                                 No
    ...
    Mean of X (Peer pressure): 3.07
    Mean of Y (Rate your academic stress index): 3.72
    Variance of X: 1.17
    Variance of Y: 1.07
```

Figure 49: **Regression Line Fit (Task 3)**
Shows the best-fit regression line: $\hat{Y} = 2.3335 + 0.4519X$, plotted over the scatter data.

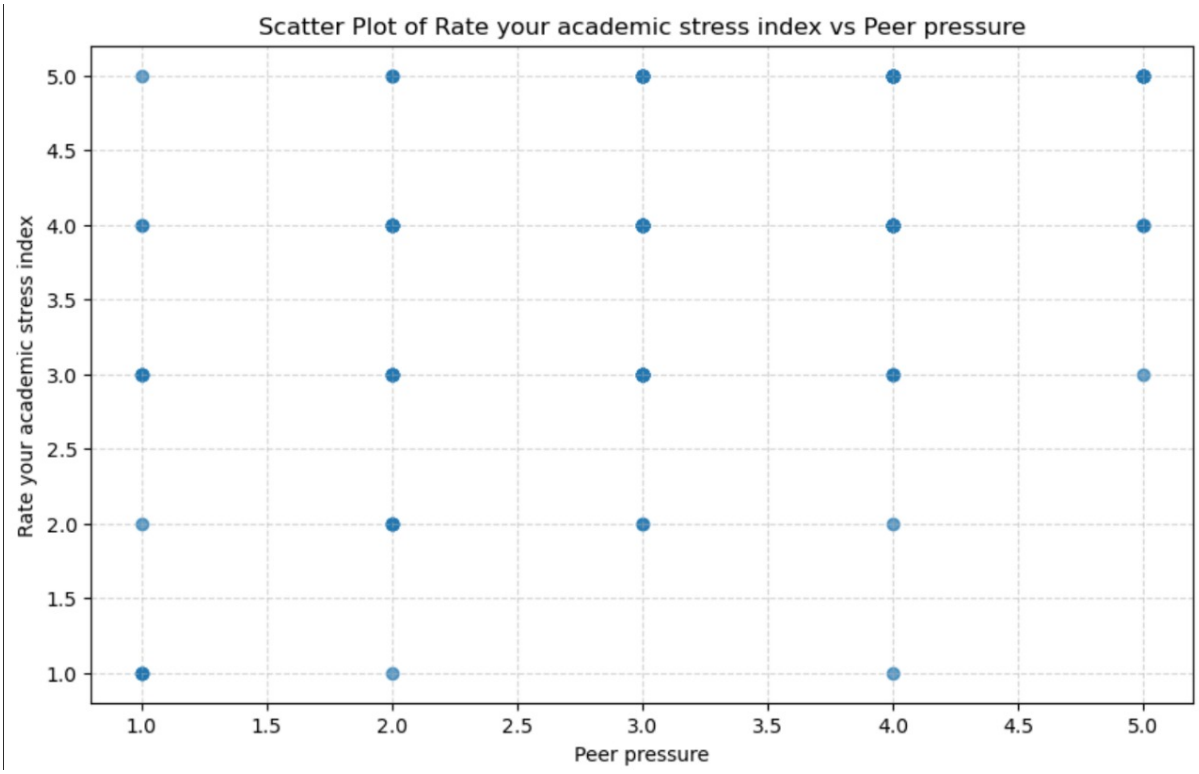
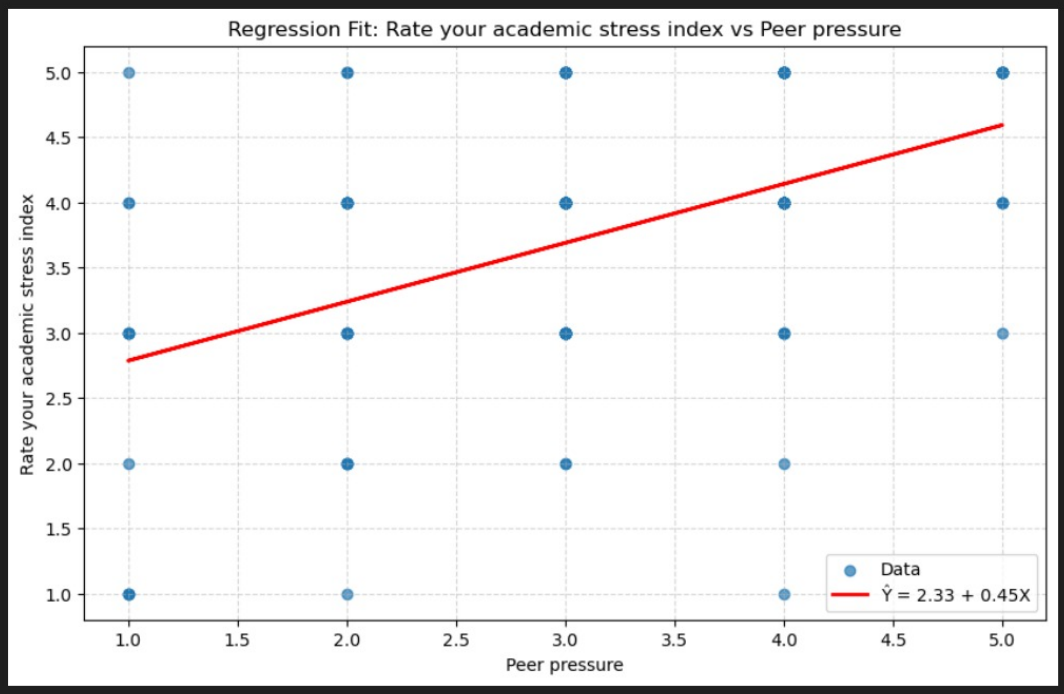



Figure 50: **Task C: Systematic Sampling Output**
Shows systematic sampling generation applied on the dataset for Milestone 06 Part C.

```
--- Task 2: Regression Parameters ---
Slope (β1): 0.4519
Intercept (β0): 2.3335
Regression Equation: Ŷ = 2.3335 + 0.4519X
```




```
 Task 4: Model Strength ---
Pearson Correlation (r): 0.4744
Coefficient of Determination (R²): 0.2251

--- OPTIONAL: Verification with scikit-learn ---
Manual β1: 0.4519 | SKL β1: 0.4519
Manual β0: 2.3335 | SKL β0: 2.3335
Manual R²: 0.2251 | SKL R²: 0.2251
```

Figure 51: **Task 4 Output: Correlation and $R^2$ Values**
Displays Pearson correlation ($r = 0.4744$) and Coefficient of Determination ($R^2 = 0.2251$), confirming a moderate positive relationship between Peer Pressure and Academic Stress.

# Reflection

In this milestone, I learned how to perform simple linear regression manually using my dataset. I selected *academic competition* as the independent variable and *academic stress index* as the dependent variable. After calculating the slope and intercept step by step, I gained a clearer understanding of how regression works behind the formulas. The scatter plot showed a positive relationship: higher academic competition is associated with higher academic stress. The correlation coefficient ($r$) and the

coefficient of determination ($R^2$) supported this by indicating both the strength of the relationship and how much variation in stress is explained by the model.
Finally, comparing the manual regression results with scikit-learn's output increased my confidence because both methods produced the same values. Overall, this milestone improved my understanding of regression interpretation and helped me see how a single factor can contribute to academic stress.

# 7 Final Conclusion

## Summary of all milestone outcome

This project successfully analyzed academic stress among undergraduate students by completing all planned milestones using appropriate statistical methods. Initially, a relevant dataset on academic stress was selected, including variables such as peer pressure, family influence, academic competition, and coping strategies. The primary objective was to measure academic stress levels and identify the major contributing factors among students.
Various sampling techniques were evaluated, and stratified sampling based on academic stage was found to be the most effective method. This confirmed the heterogeneous nature of the student population and improved the accuracy of statistical estimation.
Data visualization results indicated that academic stress levels are generally high, with most students falling into higher stress categories, highlighting academic stress as a widespread issue.
Probability distribution analysis showed that the Academic Stress Index has a high average value and does not follow a normal distribution due to skewness. Further probability and hypothesis testing revealed that more than half of the students experience high academic stress. Conditional probability analysis demonstrated that academic stage and stress level are dependent, indicating that a student's academic stage influences their stress experience.
Simple linear regression analysis established a moderate positive relationship between peer pressure and academic stress. The regression results showed that peer pressure explains a significant portion of the variation in academic stress levels. Overall, the findings emphasize that academic stress is a serious concern influenced by multiple factors, particularly peer pressure and academic competition, and highlight the need for institutional mental health support and stress management initiatives.