UNIVERSITI MALAYSIA KELANTAN

ANTSBEES

# CERTIFIED DATA ANALYSTS

## ASSIGNMENT 3

NAZREEN AGOS BIN ABDUL LATIFF

Section 1: **Core Python Programming**

1. Write a Python script to:

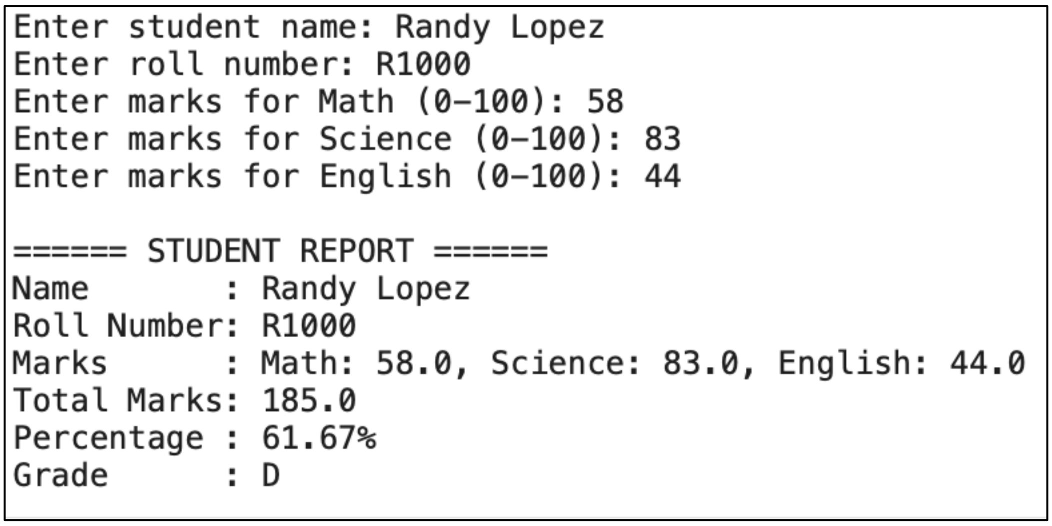
• Take student name, roll number, and mark in 3 subjects from keyboard input.
• Calculate:
o Total Marks
o Percentage
o Grade based on
▪ A: ≥ 90%
▪ B: 80–89%
▪ C: 70–79%
▪ D: 60–69%
▪ F: < 60%

2. Use appropriate data types, variables, and comments.

3. Display the result in a structured format using print () with formatting options (like sep, end).

4. Add conditional logic to:
• Print a congratulatory message if the grade is A or B.
• Warn the user if the grade is F.

**ANSWER :**

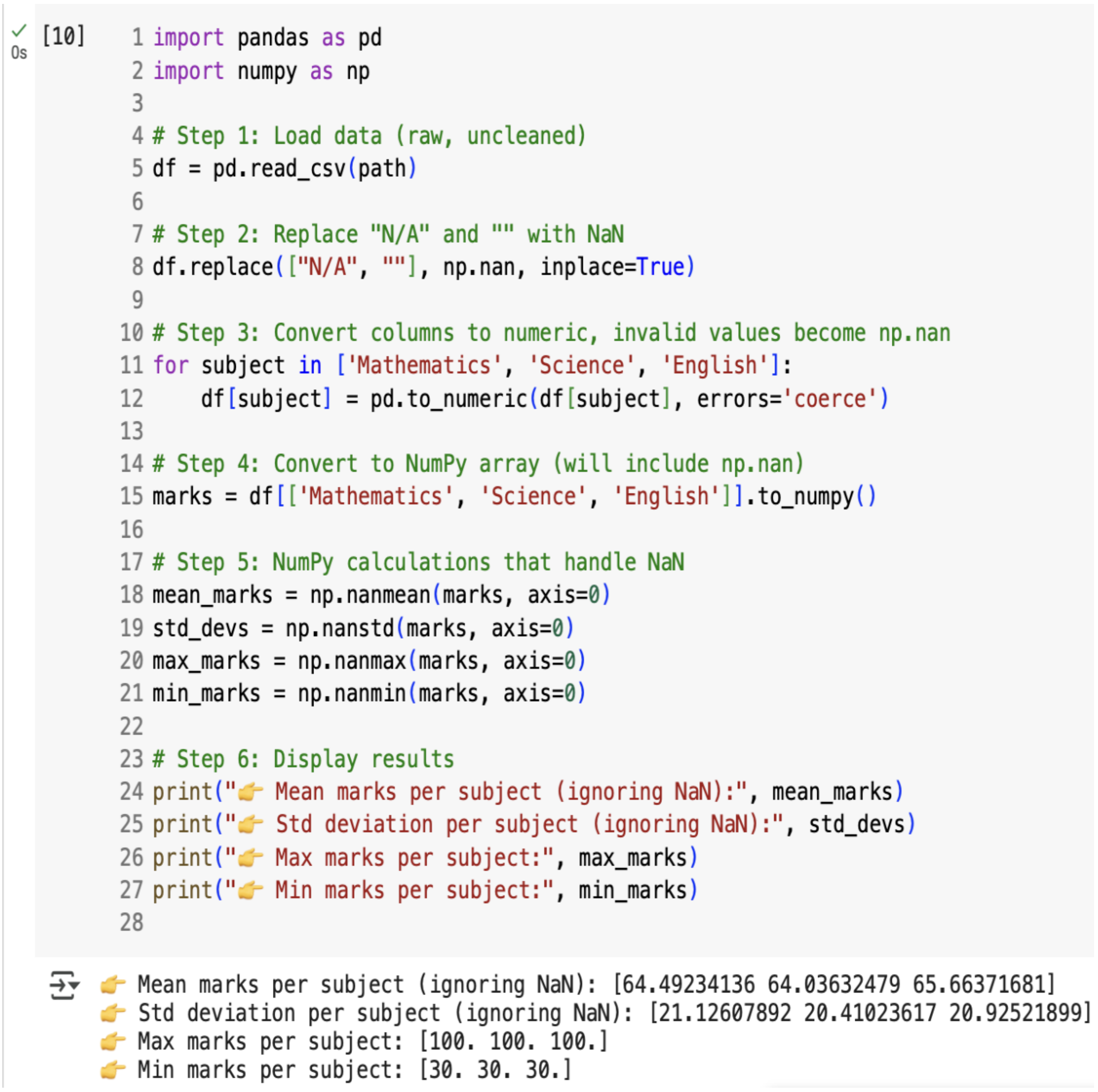


```
Enter student name: Randy Lopez
Enter roll number: R1000
Enter marks for Math (0-100): 58
Enter marks for Science (0-100): 83
Enter marks for English (0-100): 44

====== STUDENT REPORT ======
Name       : Randy Lopez
Roll Number: R1000
Marks      : Math: 58.0, Science: 83.0, English: 44.0
Total Marks: 185.0
Percentage : 61.67%
Grade      : D
```

```python
# Section 1: Student Performance Analyzer – Core Python Logic

# 1. Input: Student information
student_name = input("Enter student name: ")
roll_number = input("Enter roll number: ")

# 2. Input: Marks in three subjects
subjects = ["Math", "Science", "English"]
marks = {}

for subject in subjects:
    while True:
        try:
            score = float(input(f"Enter marks for {subject} (0–100): "))
            if 0 <= score <= 100:
                marks[subject] = score
                break
            else:
                print("Please enter a value between 0 and 100.")
        except ValueError:
            print("Invalid input. Please enter numeric marks.")

# 3. Calculations
total_marks = sum(marks.values())
percentage = total_marks / len(subjects)

# 4. Grade logic
if percentage >= 90:
    grade = 'A'
elif percentage >= 80:
    grade = 'B'
elif percentage >= 70:
    grade = 'C'
elif percentage >= 60:
    grade = 'D'
else:
    grade = 'F'

# 5. Output display
print("\n====== STUDENT REPORT ======")
print("Name       :", student_name)
print("Roll Number:", roll_number)
print("Marks      :", end=" ")
print(*[f"{sub}: {score}" for sub, score in marks.items()], sep=", ")
print("Total Marks:", total_marks)
print(f"Percentage : {percentage:.2f}%")
print("Grade      :", grade)

# 6. Conditional messaging
if grade in ['A', 'B']:
    print("🎉 Congratulations on your excellent performance!")
elif grade == 'F':
    print("⚠️ Warning: You have failed. Please consult your teacher.")
```

Section 2: **NumPy and Pandas Data Handling**

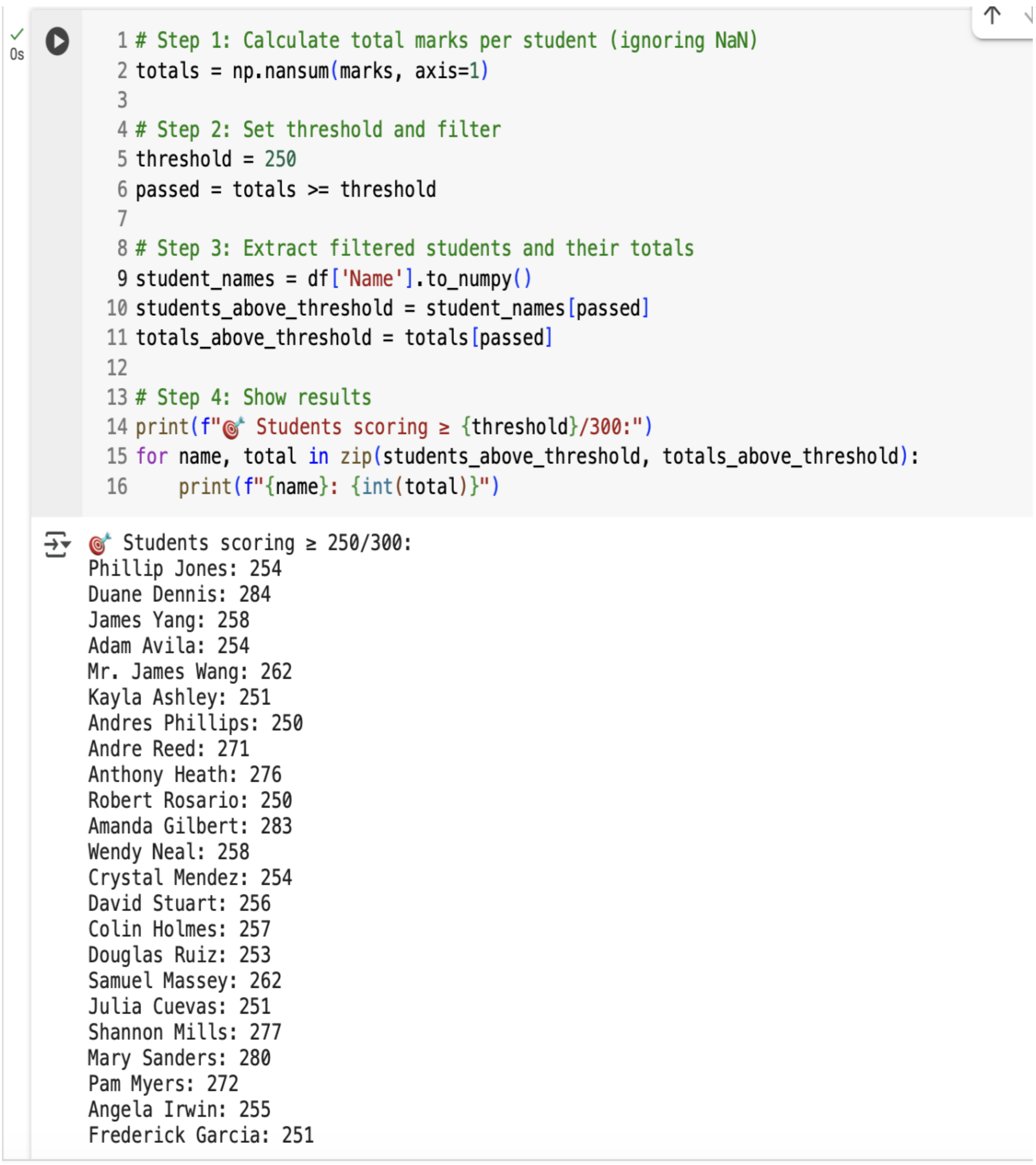
1. **Using NumPy:**
• Load student marks into NumPy arrays.
• Compute array-wise:
o Mean marks per subject
o Standard deviation
o Maximum and minimum marks.



```
[10]
import pandas as pd
import numpy as np

# Step 1: Load data (raw, uncleaned)
df = pd.read_csv(path)

# Step 2: Replace "N/A" and "" with NaN
df.replace(["N/A", ""], np.nan, inplace=True)

# Step 3: Convert columns to numeric, invalid values become np.nan
for subject in ['Mathematics', 'Science', 'English']:
    df[subject] = pd.to_numeric(df[subject], errors='coerce')

# Step 4: Convert to NumPy array (will include np.nan)
marks = df[['Mathematics', 'Science', 'English']].to_numpy()

# Step 5: NumPy calculations that handle NaN
mean_marks = np.nanmean(marks, axis=0)
std_devs = np.nanstd(marks, axis=0)
max_marks = np.nanmax(marks, axis=0)
min_marks = np.nanmin(marks, axis=0)

# Step 6: Display results
print("👉 Mean marks per subject (ignoring NaN):", mean_marks)
print("👉 Std deviation per subject (ignoring NaN):", std_devs)
print("👉 Max marks per subject:", max_marks)
print("👉 Min marks per subject:", min_marks)
```

```
👉 Mean marks per subject (ignoring NaN): [64.49234136 64.03632479 65.66371681]
👉 Std deviation per subject (ignoring NaN): [21.12607892 20.41023617 20.92521899]
👉 Max marks per subject: [100. 100. 100.]
👉 Min marks per subject: [30. 30. 30.]
```

• Filter students with total marks above a threshold (e.g., 250/300).



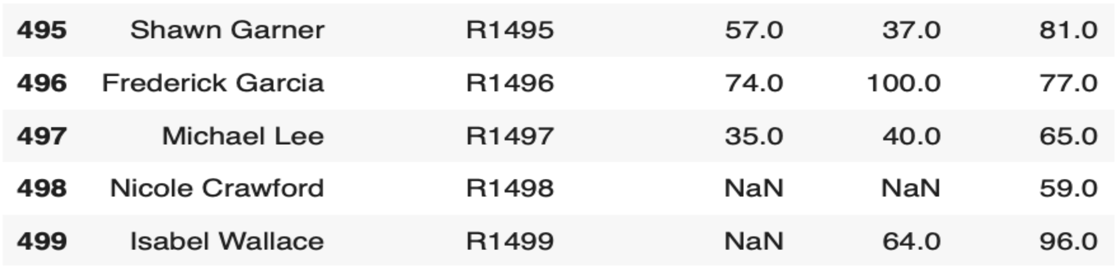
```python
# Step 1: Calculate total marks per student (ignoring NaN)
totals = np.nansum(marks, axis=1)

# Step 2: Set threshold and filter
threshold = 250
passed = totals >= threshold

# Step 3: Extract filtered students and their totals
student_names = df['Name'].to_numpy()
students_above_threshold = student_names[passed]
totals_above_threshold = totals[passed]

# Step 4: Show results
print(f"🎯 Students scoring ≥ {threshold}/300:")
for name, total in zip(students_above_threshold, totals_above_threshold):
    print(f"{name}: {int(total)}")
```

```
🎯 Students scoring ≥ 250/300:
Phillip Jones: 254
Duane Dennis: 284
James Yang: 258
Adam Avila: 254
Mr. James Wang: 262
Kayla Ashley: 251
Andres Phillips: 250
Andre Reed: 271
Anthony Heath: 276
Robert Rosario: 250
Amanda Gilbert: 283
Wendy Neal: 258
Crystal Mendez: 254
David Stuart: 256
Colin Holmes: 257
Douglas Ruiz: 253
Samuel Massey: 262
Julia Cuevas: 251
Shannon Mills: 277
Mary Sanders: 280
Pam Myers: 272
Angela Irwin: 255
Frederick Garcia: 251
```

2. **Using Pandas:**
• Load the CSV using pandas.read_csv(). Clean the data:
○ Replace missing values with 0 or appropriate estimates.
○ Convert data types as needed.

```python
# Insert CSV into Python
path="/content/drive/MyDrive/Colab Notebooks/students_raw.csv"

# Load CSV into a DataFrame
df = pd.read_csv(path)

# Just viewing without saving
pd.read_csv(path)


#----- Replace missing values with mean -------
# Replace "N/A" and empty strings with NaN
df.replace(["N/A", ""], np.nan, inplace=True)

# Convert marks columns to numeric, force errors to NaN
for subject in ['Mathematics', 'Science', 'English']:
    df[subject] = pd.to_numeric(df[subject], errors='coerce')

# Replace NaN with the mean of each subject
for subject in ['Mathematics', 'Science', 'English']:
    mean_value = df[subject].mean()
    df[subject].fillna(mean_value, inplace=True)

# Done: Data cleaned with mean replacement
print(df.tail())
```

Picture below shows the raw data #Before

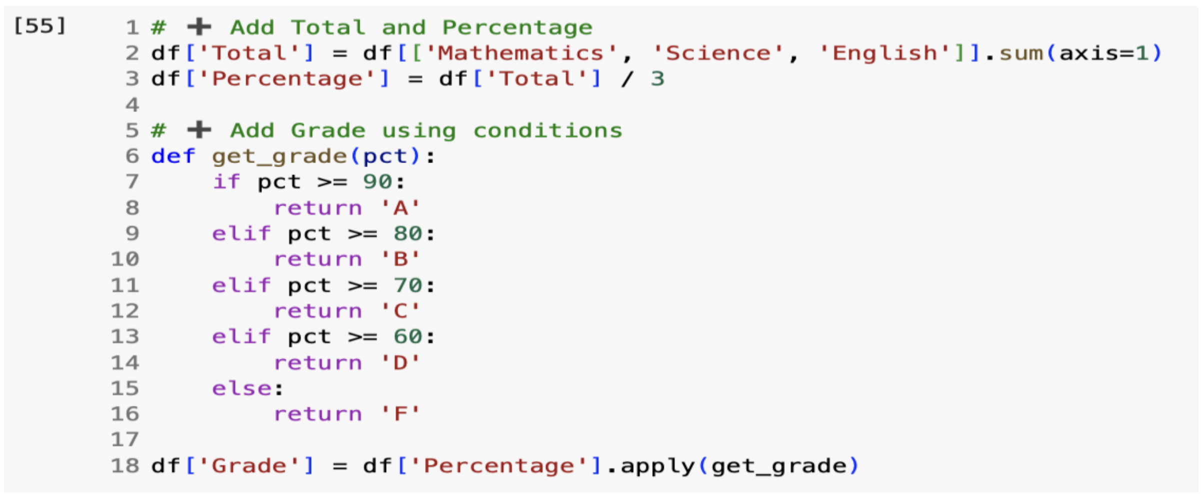


| | | | | | |
|---|---|---|---|---|---|
| **495** | Shawn Garner | R1495 | 57.0 | 37.0 | 81.0 |
| **496** | Frederick Garcia | R1496 | 74.0 | 100.0 | 77.0 |
| **497** | Michael Lee | R1497 | 35.0 | 40.0 | 65.0 |
| **498** | Nicole Crawford | R1498 | NaN | NaN | 59.0 |
| **499** | Isabel Wallace | R1499 | NaN | 64.0 | 96.0 |

After replace “N/A” with mean

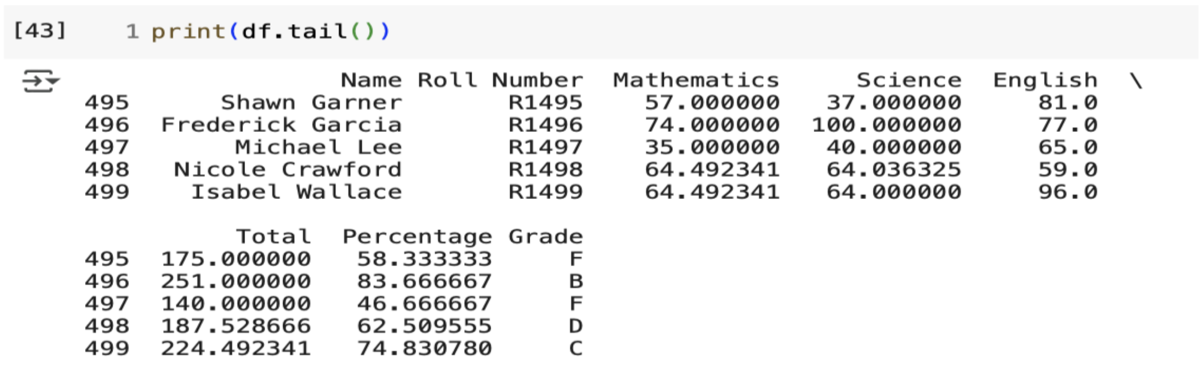
```
                 Name Roll Number  Mathematics     Science  English
495      Shawn Garner       R1495    57.000000   37.000000     81.0
496  Frederick Garcia       R1496    74.000000  100.000000     77.0
497       Michael Lee       R1497    35.000000   40.000000     65.0
498   Nicole Crawford       R1498    64.492341   64.036325     59.0
499    Isabel Wallace       R1499    64.492341   64.000000     96.0
```

• Add computed columns: Total, Percentage, Grade (use conditions)
• Save the cleaned Data Frame to a new CSV file *students_cleaned.csv*.



[55]

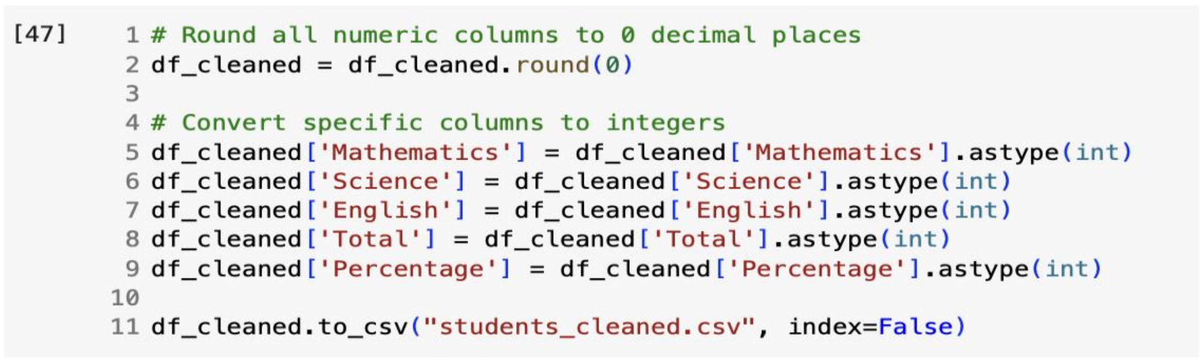
```python
# ➕ Add Total and Percentage
df['Total'] = df[['Mathematics', 'Science', 'English']].sum(axis=1)
df['Percentage'] = df['Total'] / 3

# ➕ Add Grade using conditions
def get_grade(pct):
    if pct >= 90:
        return 'A'
    elif pct >= 80:
        return 'B'
    elif pct >= 70:
        return 'C'
    elif pct >= 60:
        return 'D'
    else:
        return 'F'

df['Grade'] = df['Percentage'].apply(get_grade)
```



[43]

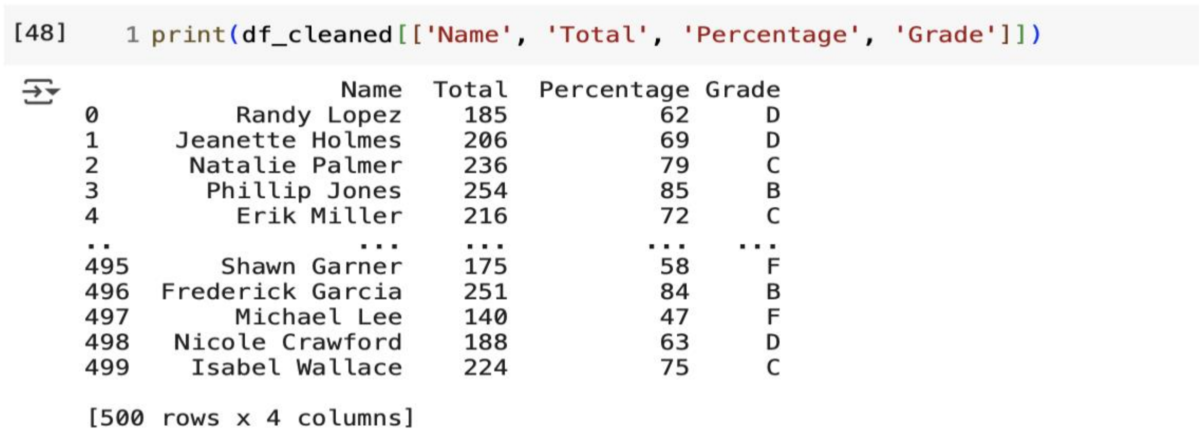
```python
print(df.tail())
```

```
                  Name Roll Number  Mathematics     Science  English  \
495       Shawn Garner       R1495    57.000000   37.000000     81.0
496   Frederick Garcia       R1496    74.000000  100.000000     77.0
497        Michael Lee       R1497    35.000000   40.000000     65.0
498    Nicole Crawford       R1498    64.492341   64.036325     59.0
499     Isabel Wallace       R1499    64.492341   64.000000     96.0

          Total  Percentage Grade
495  175.000000   58.333333     F
496  251.000000   83.666667     B
497  140.000000   46.666667     F
498  187.528666   62.509555     D
499  224.492341   74.830780     C
```



[47]

```python
# Round all numeric columns to 0 decimal places
df_cleaned = df_cleaned.round(0)

# Convert specific columns to integers
df_cleaned['Mathematics'] = df_cleaned['Mathematics'].astype(int)
df_cleaned['Science'] = df_cleaned['Science'].astype(int)
df_cleaned['English'] = df_cleaned['English'].astype(int)
df_cleaned['Total'] = df_cleaned['Total'].astype(int)
df_cleaned['Percentage'] = df_cleaned['Percentage'].astype(int)

df_cleaned.to_csv("students_cleaned.csv", index=False)
```



[48]

```python
print(df_cleaned[['Name', 'Total', 'Percentage', 'Grade']])
```

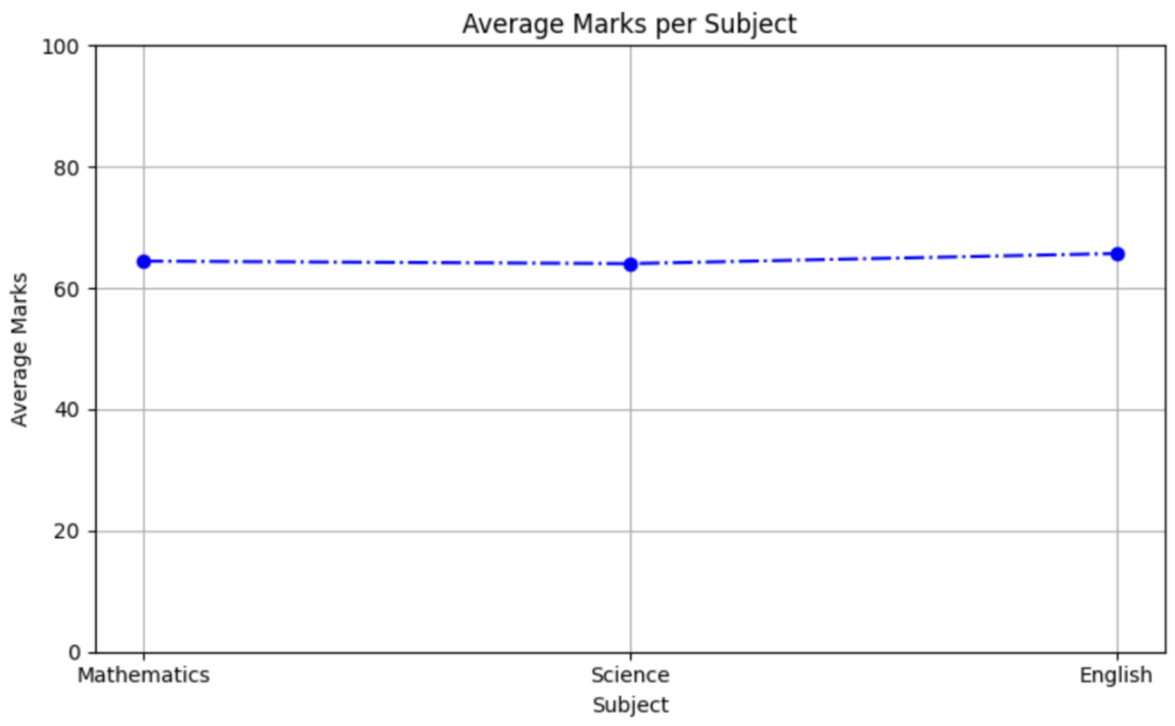
```
                  Name  Total  Percentage Grade
0          Randy Lopez    185          62     D
1      Jeanette Holmes    206          69     D
2       Natalie Palmer    236          79     C
3       Phillip Jones     254          85     B
4          Erik Miller    216          72     C
..                 ...    ...         ...   ...
495       Shawn Garner    175          58     F
496   Frederick Garcia    251          84     B
497        Michael Lee    140          47     F
498    Nicole Crawford    188          63     D
499     Isabel Wallace    224          75     C

[500 rows x 4 columns]
```

Section 3: **Data Visualization with Matplotlib**

**Line Plot:**

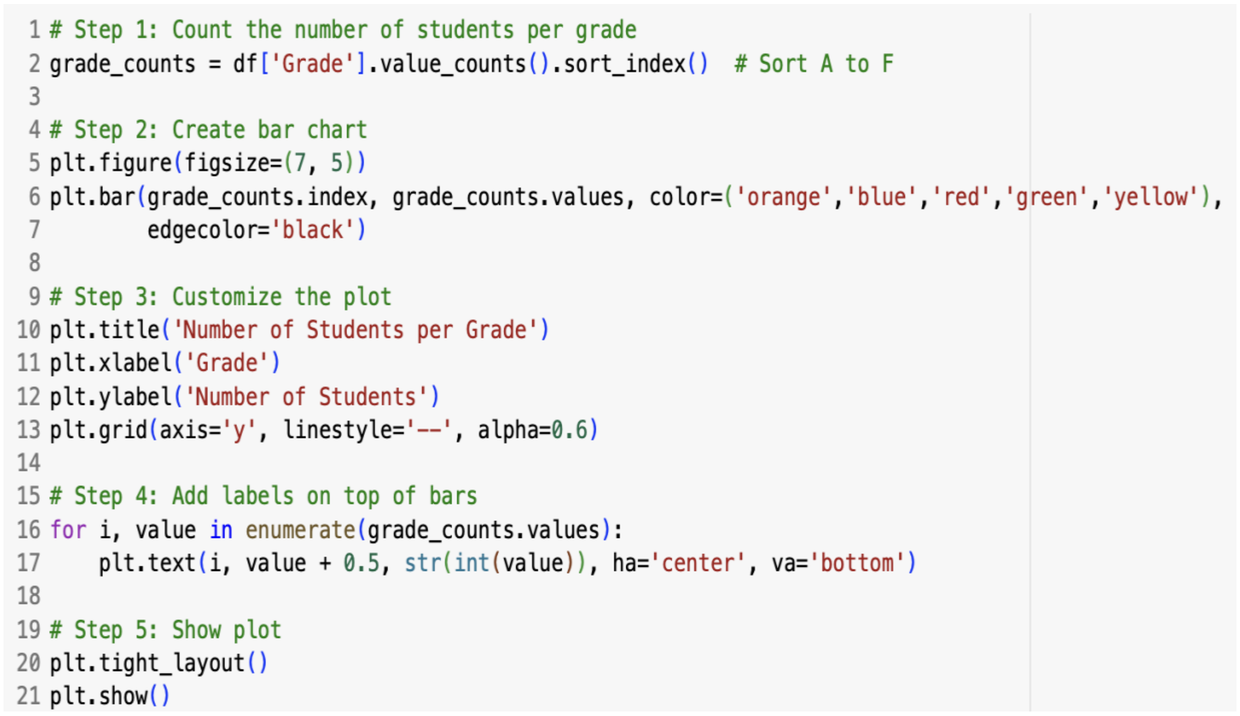
○ Show the trend of average marks in each subject

```
[5]
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Load the cleaned data
df = pd.read_csv(path)

# Step 2: Calculate average marks per subject
subjects = ['Mathematics', 'Science', 'English']
average_marks = [df[subject].mean() for subject in subjects]

# Step 3: Create line plot
plt.figure(figsize=(8, 5))
plt.plot(subjects, average_marks, marker='o', linestyle='-.', color='blue')

# Step 4: Customize the plot
plt.title('Average Marks per Subject')
plt.xlabel('Subject')
plt.ylabel('Average Marks')
plt.ylim(0, 100)  # Assuming max mark per subject is 100
plt.grid(True)

# Step 5: Show the plot
plt.tight_layout()
plt.show()
```

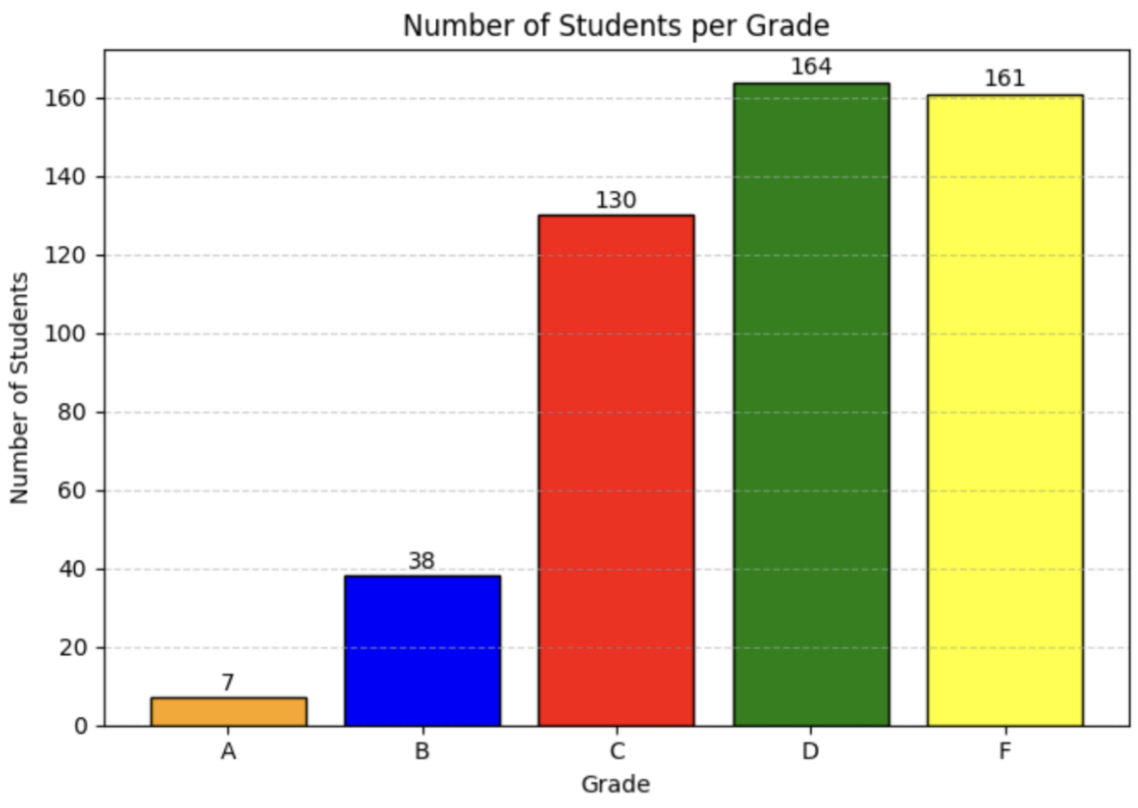


**Bar Chart:**

○ Display the number of students per grade (A, B, C, D, F)



```python
# Step 1: Count the number of students per grade
grade_counts = df['Grade'].value_counts().sort_index()  # Sort A to F

# Step 2: Create bar chart
plt.figure(figsize=(7, 5))
plt.bar(grade_counts.index, grade_counts.values, color=('orange','blue','red','green','yellow'),
        edgecolor='black')

# Step 3: Customize the plot
plt.title('Number of Students per Grade')
plt.xlabel('Grade')
plt.ylabel('Number of Students')
plt.grid(axis='y', linestyle='--', alpha=0.6)

# Step 4: Add labels on top of bars
for i, value in enumerate(grade_counts.values):
    plt.text(i, value + 0.5, str(int(value)), ha='center', va='bottom')

# Step 5: Show plot
plt.tight_layout()
plt.show()
```



**Scatter Plot:**

○ Plot percentage vs. total marks with color coding for grades.

```python
# Define color for each grade
grade_colors = {
    'A': 'orange',
    'B': 'blue',
    'C': 'red',
    'D': 'green',
    'F': 'yellow'
}

# Map colors to each row based on the grade
colors = df['Grade'].map(grade_colors)

# Create scatter plot
plt.figure(figsize=(8, 5))
plt.scatter(df['Total'], df['Percentage'], c=colors, edgecolors='black')

# Customize plot
plt.title('Percentage vs. Total Marks (Color-coded by Grade)')
plt.xlabel('Total Marks')
plt.ylabel('Percentage (%)')
plt.grid(True)

# Add legend manually
import matplotlib.patches as mpatches
legend_handles = [mpatches.Patch(color=color, label=grade) for grade, color in grade_colors.items()]
plt.legend(handles=legend_handles, title='Grade')

# Show plot
plt.tight_layout()
plt.show()
```

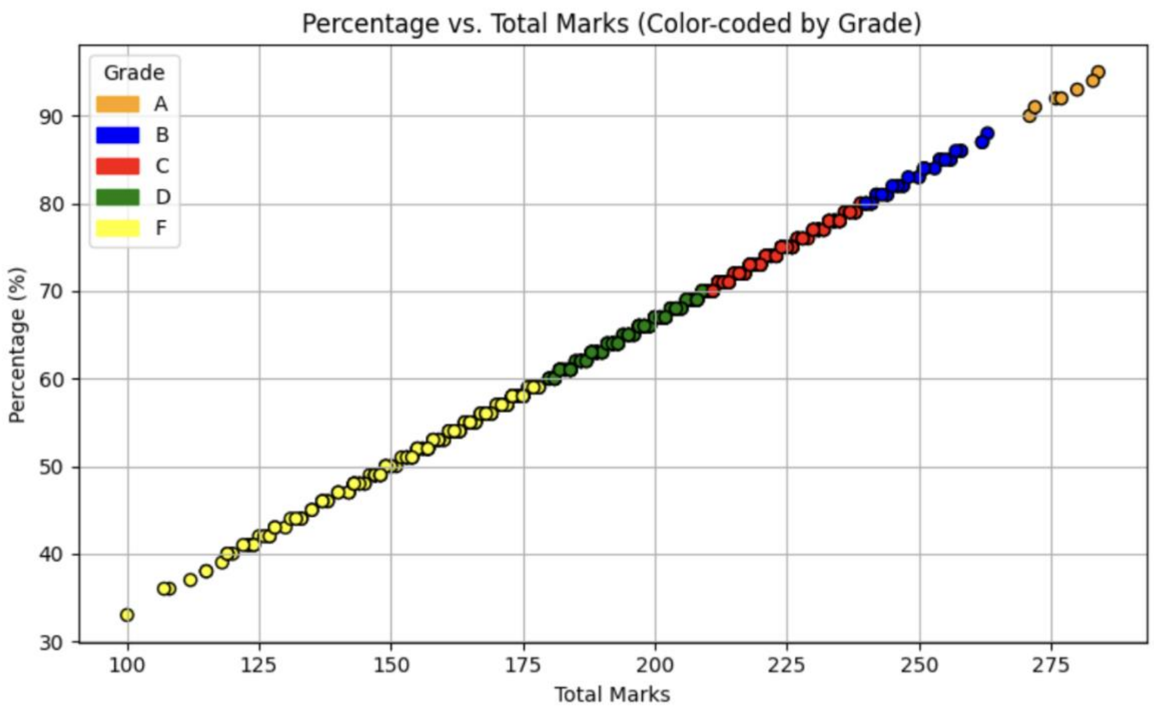
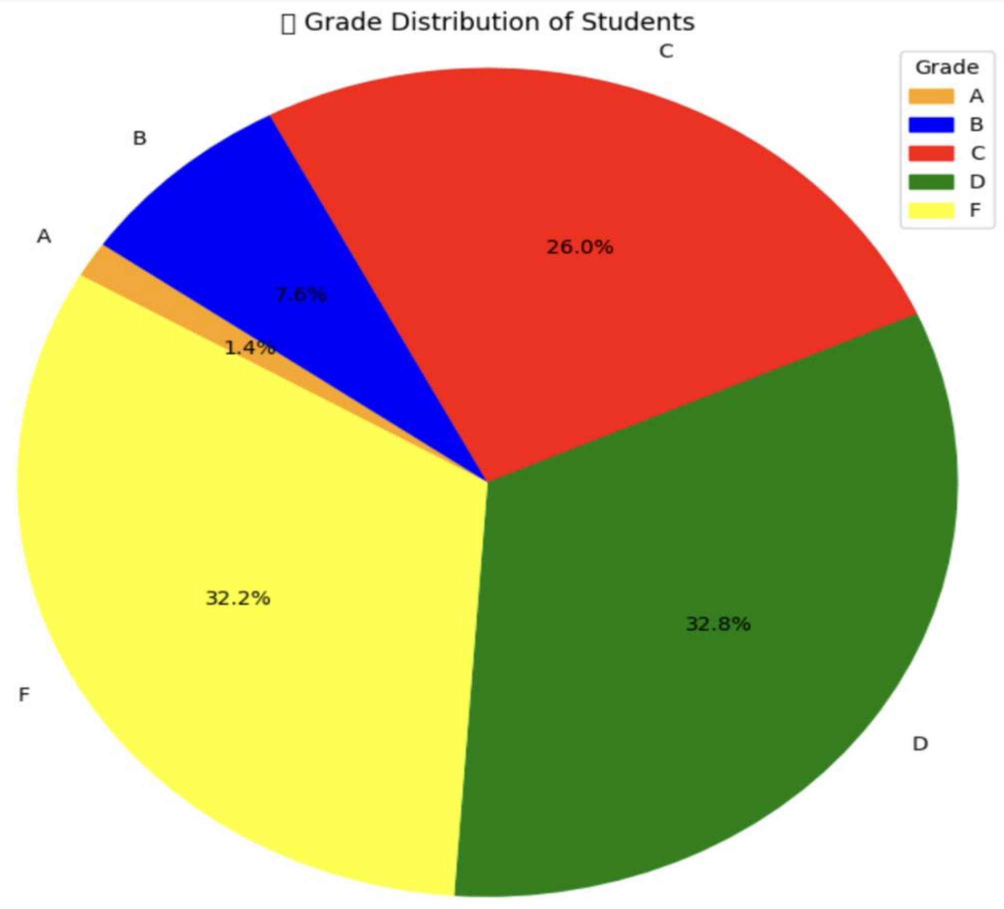


**Chart Customization:**
o Add appropriate titles, labels, legends.

Pie Chart for Grade Distribution

```python
# Count number of students per grade
grade_counts = df['Grade'].value_counts().sort_index()

# Define colors for each grade
grade_colors = {
    'A': 'orange',
    'B': 'blue',
    'C': 'red',
    'D': 'green',
    'F': 'yellow'
}
colors = [grade_colors.get(grade, 'gray') for grade in grade_counts.index]

# Create Pie Chart
plt.figure(figsize=(7, 7))
plt.pie(
    grade_counts.values,
    labels=grade_counts.index,
    colors=colors,
    autopct='%1.1f%%',
    startangle=150,
    counterclock=False
)

plt.title("🎯 Grade Distribution of Students")
plt.legend(handles=legend_handles, title='Grade')
plt.axis('equal')  # Keep circle shape
plt.tight_layout()
plt.show()
```


□ Grade Distribution of Students
Grade
A
B
C
D
F