School Exam Performance Analysis Lab

This script analyzes student test scores and attendance data using NumPy and Pandas. It demonstrates data filtering, analysis, and summarization techniques.

In [2]:
```
# Import neccessary libraries
import numpy as np
import pandas as pd
```

In [3]:
```
print("=" * 80)
print("SCHOOL EXAM PERFORMANCE ANALYSIS LAB")
print("=" * 80)
```

```
================================================================================
SCHOOL EXAM PERFORMANCE ANALYSIS LAB
================================================================================
```

In [4]:
```
# ================================================================================
# SECTION A: NumPy — Student Scores Matrix
# ================================================================================
```

In [5]:
```
# Task A1: Create the Scores Matrix
print("\n--- TASK A1: Create the Scores Matrix ---\n")
```

```
--- TASK A1: Create the Scores Matrix ---
```

In [6]:
```
# Create a 2D array with scores for 6 students across 5 subjects
# Subjects: Math, Science, English, History, Computers
scores = np.array([
    [78, 85, 88, 70, 90],   # Student 1
    [60, 65, 70, 75, 80],   # Student 2
    [90, 88, 85, 92, 95],   # Student 3
    [55, 60, 58, 62, 65],   # Student 4
    [85, 82, 80, 78, 88],   # Student 5
    [72, 75, 78, 74, 76]    # Student 6
])
```

In [7]:
```
subjects = ["Math", "Science", "English", "History", "Computers"]
students = ["S1", "S2", "S3", "S4", "S5", "S6"]
```

In [8]:
```
print("Complete Scores Matrix:")
print(scores)
print("\nSubjects:", subjects)
print("Students:", students)
```

```
Complete Scores Matrix:
[[78 85 88 70 90]
 [60 65 70 75 80]
 [90 88 85 92 95]
 [55 60 58 62 65]
 [85 82 80 78 88]
 [72 75 78 74 76]]

Subjects: ['Math', 'Science', 'English', 'History', 'Computers']
Students: ['S1', 'S2', 'S3', 'S4', 'S5', 'S6']
```

In [9]:
```python
# Task 1: Print scores for first 3 students
print("\n--- Task 1: Scores for First 3 Students ---")
print("Using slicing: scores[0:3, :]")
first_three = scores[0:3, :]
print(first_three)
print("\nExplanation: scores[0:3, :] means:")
print("  - Rows 0 to 2 (first 3 students)")
print("  - All columns (:) means all subjects")
```

```
--- Task 1: Scores for First 3 Students ---
Using slicing: scores[0:3, :]
[[78 85 88 70 90]
 [60 65 70 75 80]
 [90 88 85 92 95]]

Explanation: scores[0:3, :] means:
  - Rows 0 to 2 (first 3 students)
  - All columns (:) means all subjects
```

In [10]:
```python
# Task 2: Print all scores in English (3rd subject)
print("\n--- Task 2: All Scores in English (3rd Subject) ---")
print("Using slicing: scores[:, 2]")
english_scores = scores[:, 2]
print(english_scores)
print("\nExplanation: scores[:, 2] means:")
print("  - All rows (:) means all students")
print("  - Column 2 (3rd column, 0-indexed) is English")
```

```
--- Task 2: All Scores in English (3rd Subject) ---
Using slicing: scores[:, 2]
[88 70 85 58 80 78]

Explanation: scores[:, 2] means:
  - All rows (:) means all students
  - Column 2 (3rd column, 0-indexed) is English
```

In [11]:
```python
# Task 3: Extract scores for students who scored > 80 in Computers
print("\n--- Task 3: Students with Computers Score > 80 ---")
print("Using boolean indexing: scores[scores[:, 4] > 80]")
computers_col = scores[:, 4]  # Get all Computers scores
high_computer_scores = scores[computers_col > 80]
print("Students with Computers score > 80:")
print(high_computer_scores)
print("\nExplanation:")
print("  - scores[:, 4] gets all Computers scores")
print("  - scores[:, 4] > 80 creates a boolean mask [True/False for each student]")
print("  - scores[mask] returns only rows where mask is True")
```

```
--- Task 3: Students with Computers Score > 80 ---
Using boolean indexing: scores[scores[:, 4] > 80]
Students with Computers score > 80:
[[78 85 88 70 90]
 [90 88 85 92 95]
 [85 82 80 78 88]]

Explanation:
  - scores[:, 4] gets all Computers scores
  - scores[:, 4] > 80 creates a boolean mask [True/False for each student]
  - scores[mask] returns only rows where mask is True
```

In [12]:
```python
# Task 4: Extract students with average score > 75
print("\n--- Task 4: Students with Average Score > 75 ---")
print("Using row-wise mean and boolean indexing")
row_averages = np.mean(scores, axis=1)  # axis=1 means calculate mean across columns (
print("Average scores per student:", row_averages)
high_avg_students = scores[row_averages > 75]
print("\nStudents with average > 75:")
print(high_avg_students)
print("\nExplanation:")
print("  - np.mean(scores, axis=1) calculates average for each row (student)")
print("  - row_averages > 75 creates a boolean mask")
print("  - scores[mask] returns only students with average > 75")
```

```
--- Task 4: Students with Average Score > 75 ---
Using row-wise mean and boolean indexing
Average scores per student: [82.2 70.  90.  60.  82.6 75. ]

Students with average > 75:
[[78 85 88 70 90]
 [90 88 85 92 95]
 [85 82 80 78 88]]

Explanation:
  - np.mean(scores, axis=1) calculates average for each row (student)
  - row_averages > 75 creates a boolean mask
  - scores[mask] returns only students with average > 75
```

In [14]:
```python
# ================================================================================
# Task A2: Score Summary
# ================================================================================
print("\n--- TASK A2: Score Summary ---\n")
```

```
--- TASK A2: Score Summary ---
```

In [15]:
```python
# Calculate average score per subject (column-wise)
print("1. Average Score Per Subject (Column-wise):")
subject_averages = np.mean(scores, axis=0)  # axis=0 means calculate mean down columns
for i, subject in enumerate(subjects):
    print(f"   {subject}: {subject_averages[i]:.2f}")
```

```
1. Average Score Per Subject (Column-wise):
   Math: 73.33
   Science: 75.83
   English: 76.50
   History: 75.17
   Computers: 82.33
```

In [16]:
```python
# Find highest and lowest scores per subject
print("\n2. Highest and Lowest Scores Per Subject:")
subject_max = np.max(scores, axis=0)
subject_min = np.min(scores, axis=0)
for i, subject in enumerate(subjects):
    print(f"   {subject}: Max={subject_max[i]}, Min={subject_min[i]}")
```

```
2. Highest and Lowest Scores Per Subject:
   Math: Max=90, Min=55
   Science: Max=88, Min=60
   English: Max=88, Min=58
   History: Max=92, Min=62
   Computers: Max=95, Min=65
```

In [17]:
```python
# Find which subject had the highest overall average
print("\n3. Subject with Highest Overall Average:")
highest_avg_idx = np.argmax(subject_averages)
highest_avg_subject = subjects[highest_avg_idx]
highest_avg_value = subject_averages[highest_avg_idx]
print(f"   {highest_avg_subject} with average score of {highest_avg_value:.2f}")
```

```
3. Subject with Highest Overall Average:
   Computers with average score of 82.33
```

In [19]:
```python
# ================================================================================
# SECTION B: Pandas Series – Attendance Analysis
# ================================================================================
print("\n" + "=" * 80)
print("SECTION B: Pandas Series – Attendance Analysis")
print("=" * 80)
```

```
================================================================================
SECTION B: Pandas Series – Attendance Analysis
================================================================================
```

In [20]:
```python
# Task B1: Creating a Series
print("\n--- TASK B1: Creating a Series ---\n")

# Create a Pandas Series with attendance data
attendance = pd.Series(
    [92, 80, 95, 70, 88, 85],
    index=["S1", "S2", "S3", "S4", "S5", "S6"],
    name="Attendance (%)"
)

print("Attendance Data (as Pandas Series):")
print(attendance)
print("\nExplanation:")
print("  - Values: [92, 80, 95, 70, 88, 85] are attendance percentages")
print("  - Index: [S1, S2, S3, S4, S5, S6] are student labels")
print("  - Labels make it easier to reference data than numeric indices")
```

```
--- TASK B1: Creating a Series ---

Attendance Data (as Pandas Series):
S1    92
S2    80
S3    95
S4    70
S5    88
S6    85
Name: Attendance (%), dtype: int64

Explanation:
  - Values: [92, 80, 95, 70, 88, 85] are attendance percentages
  - Index: [S1, S2, S3, S4, S5, S6] are student labels
  - Labels make it easier to reference data than numeric indices
```

In [21]:
```python
# Task 1: Find students with attendance below 85%
print("\n--- Task 1: Students with Attendance Below 85% ---")
print("Using boolean indexing: attendance[attendance < 85]")
below_85 = attendance[attendance < 85]
print(below_85)
print("\nExplanation:")
print("  - attendance < 85 creates a boolean Series [True/False for each student]")
print("  - attendance[mask] returns only students where mask is True")
```

```
--- Task 1: Students with Attendance Below 85% ---
Using boolean indexing: attendance[attendance < 85]
S2    80
S4    70
Name: Attendance (%), dtype: int64

Explanation:
  - attendance < 85 creates a boolean Series [True/False for each student]
  - attendance[mask] returns only students where mask is True
```

In [22]:
```python
# Task 2: Count students with attendance >= 90%
print("\n--- Task 2: Count Students with Attendance >= 90% ---")
above_90 = attendance[attendance >= 90]
count_above_90 = len(above_90)
print(f"Students with attendance >= 90%: {above_90.to_dict()}")
print(f"Total count: {count_above_90}")
```

```
--- Task 2: Count Students with Attendance >= 90% ---
Students with attendance >= 90%: {'S1': 92, 'S3': 95}
Total count: 2
```

In [23]:
```python
# Task 3: Calculate average attendance
print("\n--- Task 3: Average Attendance ---")
avg_attendance = attendance.mean()
print(f"Average attendance: {avg_attendance:.2f}%")
```

```
--- Task 3: Average Attendance ---
Average attendance: 85.00%
```

In [24]:
```python
# ============================================================================
# Task B2: Combining Series with NumPy Results
# ============================================================================
```

```
print("\n--- TASK B2: Combining Series with NumPy Results ---\n")
```

```
--- TASK B2: Combining Series with NumPy Results ---
```

In [26]:
```
# Step 1: Compute average score per student from NumPy array
print("1. Computing Average Score Per Student:")
student_averages = np.mean(scores, axis=1)
print("   Student averages:", student_averages)

# Step 2: Create a Pandas Series with same student labels
print("\n2. Creating Pandas Series with Student Labels:")
avg_scores_series = pd.Series(
    student_averages,
    index=["S1", "S2", "S3", "S4", "S5", "S6"],
    name="Average Score"
)
print(avg_scores_series)

# Step 3: Identify students with both conditions:
#         - Average score > 75
#         - Attendance >= 85%
print("\n3. Students Meeting BOTH Criteria:")
print("   Criteria: Average Score > 75 AND Attendance >= 85%\n")

# Create boolean masks for both conditions
high_score_mask = avg_scores_series > 75
high_attendance_mask = attendance >= 85

# Combine masks using & (AND operator)
both_criteria = high_score_mask & high_attendance_mask

print("Students meeting both criteria:")
qualifying_students = pd.DataFrame({
    'Average Score': avg_scores_series[both_criteria],
    'Attendance (%)': attendance[both_criteria]
})
print(qualifying_students)

print("\nExplanation:")
print("  - high_score_mask = avg_scores_series > 75")
print("  - high_attendance_mask = attendance >= 85")
print("  - both_criteria = high_score_mask & high_attendance_mask")
print("  - The & operator combines boolean masks (AND logic)")

# ============================================================================
# SUMMARY AND INSIGHTS
# ============================================================================

print("\n" + "=" * 80)
print("SUMMARY AND INSIGHTS")
print("=" * 80)

print("\nKey Findings:")
print(f"1. Best performing subject: {highest_avg_subject} (avg: {highest_avg_value:.2f
print(f"2. Students with excellent performance (avg > 75 AND attendance >= 85%): {len(
print(f"3. Overall class average attendance: {avg_attendance:.2f}%")
```

```
print(f"4. Students needing improvement (avg <= 75): {len(avg_scores_series[avg_scores

print("\n" + "=" * 80)
print("END OF ANALYSIS")
print("=" * 80)
```

```
1. Computing Average Score Per Student:
   Student averages: [82.2 70.  90.  60.  82.6 75. ]

2. Creating Pandas Series with Student Labels:
S1    82.2
S2    70.0
S3    90.0
S4    60.0
S5    82.6
S6    75.0
Name: Average Score, dtype: float64

3. Students Meeting BOTH Criteria:
   Criteria: Average Score > 75 AND Attendance >= 85%

Students meeting both criteria:
    Average Score  Attendance (%)
S1           82.2              92
S3           90.0              95
S5           82.6              88

Explanation:
  - high_score_mask = avg_scores_series > 75
  - high_attendance_mask = attendance >= 85
  - both_criteria = high_score_mask & high_attendance_mask
  - The & operator combines boolean masks (AND logic)

================================================================================
SUMMARY AND INSIGHTS
================================================================================

Key Findings:
1. Best performing subject: Computers (avg: 82.33)
2. Students with excellent performance (avg > 75 AND attendance >= 85%): 3
3. Overall class average attendance: 85.00%
4. Students needing improvement (avg <= 75): 3

================================================================================
END OF ANALYSIS
================================================================================
```

In [ ]: