NumPy Use Cases for Practice(AML, CS, BDA): Raghudathesh G P, Prathviraj N

1. Sales Data Analysis

Use Case: Analyze sales data for trends, customer segmentation, or performance.
Operations:

- Count total sales per product category.
- Calculate the total revenue generated by each sales representative.
- Find the product with the highest sales.
- Group data by sales regions and calculate average sales.

---

2. Employee Data Analysis

Use Case: Manage and analyze employee-related data like salaries, departments, and performance.
Operations:

- Count the number of employees per department.
- Find the employee with the highest salary.
- Calculate average salary per department.
- Sort employees based on their performance score or salary.

---

3. Student Grades Analysis

Use Case: Analyze academic performance of students across subjects.
Operations:

- Count the number of students who passed each subject.
- Calculate the average marks per subject.
- Find the student with the highest total marks.
- Determine the top 3 students based on average marks.

---

4. Weather Data Analysis

Use Case: Work with temperature/rainfall data to identify patterns.
Operations:

- Find the maximum and minimum temperature recorded each month.
- Calculate average rainfall per season.
- Count how many days had temperature above a threshold (e.g., >35°C).
- Identify the month with the highest average temperature.

---

5. Stock Market Data Analysis

Use Case: Analyze stock prices for investment insights.
Operations:

- Calculate daily returns for each stock.
- Find the stock with the highest average return.
- Count how many days each stock had a positive return.
- Determine the maximum drawdown (largest fall from peak price).

---

6. Hospital Patient Data Analysis

Use Case: Manage and analyze patient records for healthcare insights.
Operations:

- Count the number of patients per disease category.
- Find the patient with the longest hospital stay.
- Calculate average age of patients per department.
- Identify the most common diagnosis.

---

7. E-commerce Website Traffic Analysis

Use Case: Study website visitor data for patterns.
Operations:

- Count the number of visitors per day/week.
- Find the day with the highest number of visitors.
- Calculate average time spent by users on the website.
- Determine bounce rate (users who visited only 1 page).

---

8. Transportation Data Analysis

Use Case: Analyze public transport usage across routes and timings.
Operations:

- Count the number of passengers per route.
- Calculate average occupancy per bus/train.
- Find the route with maximum passengers.
- Identify peak hours of usage.

---

9. Retail Store Inventory Analysis

Use Case: Manage and analyze product stock levels.
Operations:

- Count the total number of products in stock.
- Find the product with the lowest inventory.
- Calculate average stock per product category.
- Identify products that need restocking (below threshold).

---

10. Sports Team Performance Analysis

Use Case: Analyze performance of players and teams across matches.
Operations:

- Count the number of wins per team.
- Find the player with the highest average score.
- Calculate total goals/runs/points scored by each team.
- Determine the most consistent performer based on statistics.

NumPy 3D Array Transpose – Easy Problems

Problem 1: Full Axis Permutation

Create a 3D array of shape `(2, 3, 4)`.

- Transpose it so that the shape becomes `(4, 2, 3)`.
- Verify by printing both shapes.
- Check that element `[0, 1, 2]` in the original corresponds to `[2, 0, 1]` in the transposed array.

Problem 2: From (Batch, Channels, Height, Width) → (Batch, Height, Width, Channels)

In deep learning, images are often stored as `(2, 3, 28, 28)` → 2 images, 3 channels, 28×28 pixels.

- Transpose it to shape `(2, 28, 28, 3)`.
- Verify the new layout by checking shapes.
- Confirm that element `[1, 0, 10, 15]` in original moves to `[1, 10, 15, 0]` in transposed.

Problem 3: Transpose Twice

Create a 3D array of shape `(3, 4, 5)`.

- First, transpose it to `(5, 3, 4)`.
- Then transpose the result back to `(3, 4, 5)` using another `transpose`.
- Confirm that the final array is identical to the original (`np.array_equal`).

Problem 4: Multi-Step Swap

You have weather data stored as `(Years, Months, Days)` with shape `(2, 12, 30)`.

- Transpose it so that the shape becomes `(30, 12, 2)` → days first, years last.
- Find the value at `[year=1, month=5, day=10]` in the original and confirm its new position in the transposed array.

Problem 5: Combining Transpose with Reshape

Create a 3D array of shape `(2, 3, 4)`.

- First, transpose it to `(3, 2, 4)`.
- Then reshape the result into a 2D array of shape `(6, 4)`.
- Print both intermediate and final shapes.
- Verify that the reshaping preserves the total number of elements.

NumPy Transpose Problems with 3D Arrays (Use Case Based)

Problem 1: Medical Imaging Data

A hospital stores MRI scans in the format `(patients, slices, pixels)`.

- Create a NumPy array of shape `(2, 4, 5)` representing **2 patients**, each with **4 slices** of size `5 pixels`.
- Transpose the data so that the shape becomes `(4, 2, 5)` → now each slice groups data from both patients.
- Verify the new shape and check where element `[patient=1, slice=2, pixel=3]` is located in the new layout.

Problem 2: Video Frame Data

A video dataset is stored in the format `(frames, height, width)`.

- Create a NumPy array of shape `(6, 3, 4)` → **6 frames**, each with **3x4 pixels**.
- Transpose it to `(3, 4, 6)` → now each pixel position shows its value across all frames.
- Verify that frame-wise data is correctly rearranged by checking positions before and after transpose.

Problem 3: IoT Sensor Data

A smart building records temperature with **3 sensors** every hour for **2 days**. Data is stored as `(days, hours,`

`sensors)`.

- Create a NumPy array of shape `(2, 4, 3)` → **2 days**, **4 hours/day**, **3 sensors**.
- Transpose it to `(3, 2, 4)` so that the first axis corresponds to **sensors**, making analysis sensor-wise.
- Verify the mapping of element `[day=1, hour=2, sensor=0]`.

Problem 4: Sports Team Performance

A league records match statistics for **2 teams** across **3 matches**, tracking **4 performance metrics** per match. Data is stored as `(teams, matches, metrics)`.

- Create a NumPy array of shape `(2, 3, 4)`.
- Transpose it to `(3, 2, 4)` → now each match groups the performance of both teams.
- Confirm where the stats of **Team 1, Match 2, Metric 3** go in the new layout.

Problem 5: Climate Data

A research project stores temperature readings for **2 years**, each with **3 months**, and **5 cities**. Format: `(years, months, cities)`.

- Create a NumPy array of shape `(2, 3, 5)`.
- Transpose it to `(5, 2, 3)` → so each city has its own block of data across years and months.
- Check that the total number of elements remains the same after transpose.

NumPy Fancy Indexing Problems (1D, 2D, 3D)

1D Array Use Cases

Problem 1: Stock Prices

Create a 1D NumPy array of **30 stock prices**.

- Extract the prices on **prime-numbered days**.
- Find the **top 5 highest prices** using fancy indexing.

Problem 2: Exam Scores

Generate a 1D array of **50 student scores** (0–100).

- Extract the scores of the **top 10 students** (highest scores).
- Replace scores below 40 with `-1` using fancy indexing.

Problem 3: Sensor Fault Detection

Create a 1D array of **100 sensor readings** (random floats).

- Find all indices where values are **above mean + 1 std deviation**.
- Extract those faulty readings using fancy indexing.

Problem 4: DNA Sequence Encoding

Represent a DNA sequence (A=0, C=1, G=2, T=3) as a 1D NumPy array of length 20.

- Extract all occurrences of `G` and `T`.
- Replace all `A` with `-1` using fancy indexing.

Problem 5: Employee Salaries

Create a 1D array of **20 salaries**.

- Extract salaries of employees whose index is in `[2, 5, 7, 11, 15]`.
- Give a **10% raise** only to these extracted salaries using fancy indexing.

2D Array Use Cases

Problem 6: Matrix Row/Column Extraction

Create a `6x6` matrix of integers from 1 to 36.

- Extract all elements from **rows [1, 3, 5]** and **columns [0, 2, 4]** using fancy indexing.

Problem 7: Student Marks Table

Generate a `10x5` array representing **10 students** and **5 subjects**.

- Extract marks of **students [2, 4, 6, 8]** in **subjects [1, 3]**.
- Replace all marks below 30 with `0` for these students.

Problem 8: Airline Seat Booking

Create a `12x6` seating matrix (rows × columns).

- Extract all **aisle seats** (first and last column).
- Mark **seats [row=2, cols=[1,2,4]]** as booked (`-1`) using fancy indexing.

Problem 9: Chessboard Pattern

Create an $8x8$ matrix with values from 0–63.

- Extract all **black squares** (even row + odd column, odd row + even column).
- Replace them with -1.

Problem 10: Product Sales Data

Create a $5x7$ array representing sales of **5 products** over **7 days**.

- Extract sales of products $[1, 3]$ on days $[2, 4, 6]$.
- Increase those extracted values by 20%.

3D Array Use Cases

Problem 11: Video Frames (RGB)

Create a $4x3x3$ array representing **4 frames**, each $3x3$ pixel.

- Extract the **center pixel** from each frame using fancy indexing.

Problem 12: Weather Data

Create a $(2, 12, 30)$ array → **2 years**, **12 months**, **30 days**.

- Extract temperatures of **months [0, 5, 11]** for **year 1** on **days [10, 20]**.

Problem 13: Hospital Patient Records

Create a $(3, 4, 5)$ array → **3 patients**, **4 days**, **5 test results/day**.

- Extract test results of **patients [0, 2]** on **day 3** for **tests [1, 3]**.

Problem 14: Sports Team Analytics

Create a $(5, 6, 4)$ array → **5 teams**, **6 matches**, **4 stats each**.

- Extract stats of **teams [1, 3, 4]** for **matches [2, 5]**.

Problem 15: Movie Ratings

Create a `(10, 5, 4)` array → **10 movies**, **5 critics**, **4 rating categories**.

- Extract ratings of **movies [2, 4, 7]** by **critics [1, 3]** in **categories [0, 2]**.

Problem 16: IoT Device Monitoring

Create a `(3, 24, 7)` array → **3 devices**, **24 hours**, **7 days**.

- Extract hourly readings for **devices [0, 2]** on **days [1, 3, 5]**.

Problem 17: Retail Store Inventory

Create a `(4, 10, 6)` array → **4 stores**, **10 products**, **6 months**.

- Extract data for **stores [1, 3]**, **products [2, 5, 7]**, and **months [0, 2, 4]**.

Problem 18: Scientific Experiment

Create a `(2, 5, 8)` array → **2 trials**, **5 experiments**, **8 observations each**.

- Extract **trial 1** results for **experiments [1, 4]** at **observations [2, 5, 7]**.

Problem 19: E-commerce Orders

Create a `(6, 4, 3)` array → **6 customers**, **4 orders each**, **3 features/order**.

- Extract features of **customers [0, 2, 5]** for **orders [1, 3]**.

Problem 20: Image Dataset

Create a `(5, 28, 28)` array → **5 grayscale images of 28×28 pixels**.

- Extract the **corner pixels** (top-left, top-right, bottom-left, bottom-right) from each image using fancy indexing.

Detective Case Study: The Mystery of the Stolen Diamonds

Scene 1: The Crime Scene

Detective **Arjun Mehta** arrives at a jewelry store in Mumbai where diamonds have been stolen.
The store CCTV captured **100 frames of data**, each containing sensor readings.

Task:

- Create a **1D NumPy array** of 100 integers (0–9) representing sensor triggers.
- Find out how many times the sensor detected activity (`value > 5`).

Scene 2: Suspect List

Inspector **Rani Sharma** provides a list of **20 suspects** with their **ages**.
Ages are randomly assigned.

Task:

- Create a **1D array** of 20 ages.
- Use fancy indexing to extract the ages of suspects with even indices.
- Find the **youngest and oldest suspects**.

Scene 3: Witness Statement

A witness says: "I saw 5 people entering the store in pairs."
CCTV logs show movements stored as a **10x2 matrix** (10 rows, 2 values each: entry time and exit time).

Task:

- Create a **2D NumPy array** of shape (10, 2) representing entry and exit times.
- Find the **average entry time**.
- Extract all rows where exit time < entry time + 5 minutes (suspicious quick visits).

Scene 4: Car Parking Data

Outside the store, **parking lot cameras** recorded cars over 7 days and 5 time slots per day.

Task:

- Create a **2D array of shape (7, 5)** representing number of cars.
- Find the **day with maximum car traffic**.
- Extract car counts for **days [2, 4, 6]** and **time slots [1, 3]** using fancy indexing.

Scene 5: Suspect Movements

Detective Arjun obtains **GPS coordinates** of 3 suspects over 4 days, with 2 readings/day.

Task:

- Create a **3D array of shape (3, 4, 2)** → (suspects, days, coordinates).
- Transpose it to `(4, 3, 2)` to compare suspects day-wise.
- Extract locations of **suspect 2** on **days [1, 3]**.

Scene 6: Hidden Transactions

Rani uncovers financial records of **5 suspects**, each with 6 months of transactions.

Task:

- Create a **2D array of shape (5, 6)**.
- Find suspects whose **average monthly spending > 50,000**.
- Extract transactions of **suspects [1, 3]** in **months [2, 4]**.

Scene 7: Fingerprint Evidence

At the scene, police collected **3 fingerprint scans**, each stored as a **4x4 pixel grayscale image**.

Task:

- Create a **3D array of shape (3, 4, 4)**.
- Extract the **center 2x2 region** of each image.
- Replace all pixel values < 100 with 0 (noise removal).

Scene 8: Decoding the Cipher

The thieves left a coded message as numbers `[65, 66, 67, 68, 69]`.
It maps to ASCII letters.

Task:

- Create a **1D array of ASCII values**.
- Convert them into characters using vectorized operations (`chr`).
- Reveal the secret word.

Scene 9: Narrowing the Suspects

Based on data, only **4 suspects remain**.
Their data includes **height, weight, and age**, stored in a `4x3` matrix.

Task:

- Create the array and label columns as `[Height, Weight, Age]`.
- Extract suspects with **weight > 70** and **age < 30**.

Scene 10: The Final Chase

CCTV confirms that **2 cars were used in the escape**, tracked every 3 hours for 2 days.

Task:

- Create a **3D array of shape (2, 2, 8)** → (cars, days, 8 time slots).
- Transpose it to `(2, 8, 2)` → (cars, time slots, days).
- Find at which time slot both cars were parked at the **same location**.

□ The Verdict

After all the analysis, Detective **Arjun Mehta** and Inspector **Rani Sharma** discover the mastermind:

- The suspect whose **sensor activity matched**,
- **Car movement aligned**, and
- **Financial transactions spiked abnormally**.

Final Task:

- Combine your extracted suspects' data across all arrays and print the **prime suspect ID**.

Detective Story: The Mystery of the Stolen Diamonds

Author: Detective Training Exercise

Goal: Practice all major NumPy operations through a fun mystery case.

```
import numpy as np

print("□ Scene 1: The Crime Scene") print("Detective Arjun Mehta arrives at a jewelry store in Mumbai. Diamonds have been stolen!") print("The CCTV captured 100 sensor readings (values 0–9).")
```

Create 1D array of sensor data

```
sensor_data = np.random.randint(0, 10, size=100)
```

TODO: Find how many times the sensor detected activity (>5)

high_activity_count = ...

print("\nScene 1 Result → High activity count:", "???")

---

print("\n Scene 2: Suspect List") print("Inspector Rani Sharma provides a list of 20 suspects with their ages.")

ages = np.random.randint(18, 60, size=20)

TODO: Extract ages of suspects with even indices

even_index_ages = ...

TODO: Find youngest and oldest suspects

youngest = ...

oldest = ...

print("\nScene 2 Result → Youngest:", "???", "Oldest:", "???")

---

print("\n Scene 3: Witness Statement") print("Witness says: 'I saw 5 people entering the store in pairs.'") print("CCTV logs are stored as a 10x2 matrix (entry, exit times).")

cctv_logs = np.random.randint(0, 24, size=(10, 2))

TODO: Average entry time

avg_entry = ...

TODO: Extract rows where exit < entry+5

quick_visits = ...

print("\nScene 3 Result → Suspicious visits:\n", "???")

---

print("\n Scene 4: Car Parking Data") print("Parking lot cameras recorded cars for 7 days, 5 time slots each.")

cars = np.random.randint(0, 20, size=(7, 5))

TODO: Find day with maximum car traffic

max_day = ...

TODO: Extract traffic for days [2,4,6] and time slots [1,3]

subset_traffic = ...

print("\nScene 4 Result → Max traffic day:", "???")

---

print("\n Scene 5: Suspect Movements") print("GPS coordinates of 3 suspects, 4 days, 2 coords/day.")

gps = np.random.randint(0, 100, size=(3, 4, 2))

TODO: Transpose to (4,3,2)

gps_transposed = ...

TODO: Extract suspect 2 on days [1,3]

suspect2_movements = ...

print("\nScene 5 Result → Movements of suspect 2:\n", "???")

---

print("\n Scene 6: Hidden Transactions") print("Financial records of 5 suspects, 6 months.")

transactions = np.random.randint(20000, 100000, size=(5, 6))

TODO: Find suspects with avg spending > 50,000

big_spenders = ...

TODO: Extract suspects [1,3] months [2,4]

special_tx = ...

print("\nScene 6 Result → Big spenders:", "???")

---

print("\n Scene 7: Fingerprint Evidence") print("3 fingerprint scans as 4x4 images.")

fingerprints = np.random.randint(0, 255, size=(3, 4, 4))

TODO: Extract center 2x2 region of each image

centers = ...

TODO: Replace pixels < 100 with 0

fingerprints_clean = ...

print("\nScene 7 Result → Cleaned fingerprints:\n", "???")

---

print("\n Scene 8: Decoding the Cipher") print("A coded message: [65, 66, 67, 68, 69]")

cipher = np.array([65, 66, 67, 68, 69])

TODO: Convert to characters

message = ...

print("\nScene 8 Result → Secret message:", "???")

---

print("\n Scene 9: Narrowing the Suspects") print("Remaining 4 suspects: Height, Weight, Age")

suspects_data = np.random.randint(20, 100, size=(4, 3))

TODO: Extract suspects with weight > 70 and age < 30

filtered_suspects = ...

print("\nScene 9 Result → Filtered suspects:\n", "???")

---

print("\n Scene 10: The Final Chase") print("2 cars tracked every 3 hours for 2 days → shape (2,2,8)")

cars_tracking = np.random.randint(0, 50, size=(2, 2, 8))

TODO: Transpose to (2,8,2)

cars_transposed = ...

TODO: Find times both cars share same location

same_location_times = ...

print("\nScene 10 Result → Same location times:", "???")

---

print("\n Final Verdict") print("Combine results to reveal the PRIME SUSPECT!")

TODO: Combine logic → pick suspect ID based on patterns

prime_suspect = ...

print("\n The Prime Suspect is:", "???")
