📘 Lab Report

Name: **HIDAYAT ULLAH**

R No: **B24F0050AI099**

Lab No: **03**

Date: **20/09/2025**

Submitted to: **Engineer ABDULLAH SAJID**

TASK 1: DATA VALIDATION AND CLEANING SYSTEM

`Scenario:`
You're building a data preprocessing system for an AI application. The system needs.
to validate and clean numerical data from various sensors. Your task is to create a program that:

1. Accepts a list of numerical values as input.
2. Identifies and removes outliers (values that are more than 2 standard deviations from the mean).
3. Replaces missing values (represented as None or -1) with the mean of valid values.
4. Returns the cleaned dataset and statistics about the cleaning process.
   `Input:`
   A list of numerical values, which may include outliers and missing values.

```
data = [12.5, 15.3, 11.8, None, 14.2, 100, 13.7, -1, 12.9, 14.1]
print("Original Data:", data)

valid_values = [x for x in data if x is not None and x != -1]
mean_val = sum(valid_values) / len(valid_values)

variance = sum((x - mean_val) ** 2 for x in valid_values) / len(valid_values)
std_val = variance ** 0.5

cleaned_data = []
outliers_removed = 0
missing_replaced = 0

for val in data:
    if val in (None, -1):
        cleaned_data.append(mean_val)
        missing_replaced += 1
    elif val < (mean_val - 2 * std_val) or val > (mean_val + 2 * std_val):
        outliers_removed += 1
        continue
    else:
        cleaned_data.append(val)

print("Cleaned Data:", cleaned_data)

print("Number of Outliers removed:", outliers_removed)
print("Number of missing Values replaced:", missing_replaced)
```

```
Original Data: [12.5, 15.3, 11.8, None, 14.2, 100, 13.7, -1, 12.9, 14.1]
Cleaned Data: [12.5, 15.3, 11.8, 24.3125, 14.2, 13.7, 24.3125, 12.9, 14.1]
Number of Outliers removed: 1
Number of missing Values replaced: 2
```

Descriptions

This program clean data. First it remove missing values like None or -1 and replace them with mean. Then it calculate mean and standard deviation. Outliers more than 2 std are removed. At the end program return clean data with mean and std.z

---

TASK 2: INVENTORY MANAGEMENT SYSTEM

`Scenario:`

You're developing an inventory management system for a warehouse. The system needs to:

1. Track products and their quantities.
2. Process transactions (add, remove, or check stock).
3. Alert when stock levels are low (below a threshold).
4. Generate a summary report.

   `Input:`

   A series of commands in the format: operation product_name quantity.

```
def inventory_management_system():
    # Initialize parallel lists for products and quantities
    products = []
    quantities = []
    low_stock_threshold = 5

    print("Inventory Management System")
    print("Commands: add [product] [quantity], remove [product] [quantity], check [product], report, exit"

    while True:
        # Get user input
        command = input("\nEnter command: ").strip().lower()

        if command == "exit":
            print("Exiting inventory system.")
            break
        elif command == "report":
            # Generate summary report
            print("\nInventory Report:")
            for i in range(len(products)):
                low_stock_msg = " (Low stock!)" if quantities[i] < low_stock_threshold else ""
                print(f"{products[i]}: {quantities[i]}{low_stock_msg}")
            continue

        # Split command into parts
        parts = command.split()
        if len(parts) < 2:
            print("Invalid command format. Use: operation product_name [quantity]")
            continue

        operation = parts[0]
        product_name = parts[1]

        # Check if product exists in inventory
        if product_name in products:
            product_index = products.index(product_name)
        else:
```

```
            product_index = -1

        # Process the operation
        if operation == "add":
            if len(parts) < 3:
                print("Please specify quantity to add.")
                continue

            try:
                quantity = int(parts[2])
                if quantity <= 0:
                    print("Quantity must be positive.")
                    continue

                if product_index == -1:
                    # Add new product
                    products.append(product_name)
                    quantities.append(quantity)
                    print(f"Added {quantity} {product_name}(s). Current stock: {quantity}")
                else:
                    # Update existing product
                    quantities[product_index] += quantity
                    print(f"Added {quantity} {product_name}(s). Current stock: {quantities[product_index]}
            except ValueError:
                print("Quantity must be a number.")

        elif operation == "remove":
            if len(parts) < 3:
                print("Please specify quantity to remove.")
                continue

            try:
                quantity = int(parts[2])
                if quantity <= 0:
                    print("Quantity must be positive.")
                    continue

                if product_index == -1:
                    print(f"Error: {product_name} not found in inventory.")
                else:
                    if quantities[product_index] < quantity:
                        print(f"Error: Not enough {product_name} in stock. Current stock: {quantities[prod
                    else:
                        quantities[product_index] -= quantity
                        print(f"Removed {quantity} {product_name}(s). Current stock: {quantities[product_i

            except ValueError:
                print("Quantity must be a number.")

        elif operation == "check":
            if product_index == -1:
                print(f"{product_name} not found in inventory.")
            else:
                print(f"Current stock of {product_name}: {quantities[product_index]}")

        else:
            print("Invalid operation. Use: add, remove, check, report, or exit.")

# Run the inventory system
if __name__ == "__main__":
    inventory_management_system()
```

```
Inventory Management System
Commands: add [product] [quantity], remove [product] [quantity], check [product], report, exit

Enter command: add laptop 5
Added 5 laptop(s). Current stock: 5

Enter command: add phone 10
Added 10 phone(s). Current stock: 10

Enter command: remove laptop 2
Removed 2 laptop(s). Current stock: 3
```

```
Enter command: check laptop
Current stock of laptop: 3

Enter command: add headphone 15
Added 15 headphone(s). Current stock: 15

Enter command: remove phone 5
Removed 5 phone(s). Current stock: 5

Enter command: check phone
Current stock of phone: 5

Enter command: report

Inventory Report:
laptop: 3 (Low stock!)
phone: 5
headphone: 15

Enter command: exit
Exiting inventory system.
```

Descriptions

This program manage warehouse. User can add, remove or check stock. If stock is less than 5, program give warning. At the end it show full report. It use lists, loops and if-else to handle stock system.

TASK 3: PASSWORD STRENGTH ANALYZER

`Scenario:`
You're building a password strength analyzer for a security system. The program should:

1. Check a password against multiple criteria.
2. Assign a strength score based on the criteria met.
3. Provide detailed feedback on how to improve the password.

   `Criteria:` • Length (at least 8 characters): +1 point.
   • Contains uppercase and lowercase letters: +1 point.
   • Contains at least one digit: +1 point.
   • Contains at least one special character: +1 point.
   • Not a common password (check against a list): +1 point.
   `Input:`
   A password string: MyP@ssw0rd.

```
password = input("Enter your password: ")
score = 0

common_pass = ['12345678', 'hello123', 'abc12345', 'qwerty', 'password']
special = "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"

print("\nPassword Strength Analysis:")

# 1. Length check
if len(password) >= 8:
    print("Length: ✓ (8+ characters)")
    score += 1
else:
    print("Length: ✗ (Too short)")

# 2. Uppercase + Lowercase
if any(c.isupper() for c in password) and any(c.islower() for c in password):
    print("Uppercase and lowercase: ✓")
    score += 1
else:
    print("Uppercase and lowercase: ✗")

# 3. Digit check
if any(c.isdigit() for c in password):
    print("Contains digit: ✓")
    score += 1
else:
    print("Contains digit: ✗")

# 4. Special character
if any(c in special for c in password):
    print("Contains special character: ✓")
    score += 1
else:
    print("Contains special character: ✗")

# 5. Not a common password
if password not in common_pass:
    print("Not a common password: ✓")
    score += 1
else:
    print("Not a common password: ✗")

# Final result
print(f"\nStrength score: {score}/5", end=" - ")
if score == 5:
    print("Very Strong!")
elif score >= 3:
    print("Strong")
elif score >= 2:
    print("Medium")
else:
    print("Weak")
```

```
Enter your password: MyP@ssw0rd

Password Strength Analysis:
Length: ✓ (8+ characters)
Uppercase and lowercase: ✓
Contains digit: ✓
Contains special character: ✓
Not a common password: ✓

Strength score: 5/5 - Very Strong!
```

Descriptions

This program check password strength. It give score if password long, has upper+lower, digit, special, and not common. Final score decide if password is strong, medium or weak.

FULL LAB SUMMARY

In this lab I do 3 tasks using Python conditionals and loops.

In Task 1, I make data cleaning system. It replace missing values with mean and remove outliers with std formula.
. In Task 2, I make inventory system. It can add, remove, check stock and also give low stock warning.

In Task 3, I make password analyzer. It check rules like length, special character, digits and give strength result.

This lab help me to learn how to use if-else, loops, break, continue, and also work with lists and strings in real problems.