**Lab Exercise 9 – Automating Excel Tasks with OpenPyxl for RPA**

**Objective:**  
Learn how to use Python's openpyxl library to automate common tasks in Excel as part of a Robotic Process Automation (RPA) workflow.

**Prerequisites**

1. Install Python (>=3.7).
2. Install the openpyxl library:

pip install openpyxl

1. Install additional libraries for data validation or RPA-related tasks (optional):

pip install pandas

**Scenario:**

You are working as an RPA developer, and your task is to automate the following operations in an Excel file:

1. Create a new workbook and add data to it.
2. Perform basic formatting on cells.
3. Add formulas to calculate totals.
4. Apply conditional formatting based on data values.
5. Extract specific data and save it in another worksheet.
6. Save the updated workbook.

**Lab Steps**

**Step 1: Create a New Workbook and Add Data**

1. **Code:**

from openpyxl import Workbook

# Create a workbook and active worksheet

wb = Workbook()

ws = wb.active

ws.title = "Sales Data"

# Add data

headers = ["Date", "Product", "Quantity", "Price", "Total"]

data = [

["2024-01-01", "Widget A", 10, 5.0, None],

["2024-01-02", "Widget B", 15, 7.5, None],

["2024-01-03", "Widget C", 5, 12.0, None],

]

ws.append(headers)

for row in data:

ws.append(row)

wb.save("sales\_data.xlsx")

print("Workbook created and data added!")

**Step 2: Format the Worksheet**

1. **Add bold headers and adjust column widths.**

from openpyxl.styles import Font

# Apply bold font to headers

for cell in ws[1]:

cell.font = Font(bold=True)

# Adjust column widths

column\_widths = [12, 15, 10, 10, 12]

for i, width in enumerate(column\_widths, start=1):

ws.column\_dimensions[chr(64 + i)].width = width

wb.save("sales\_data.xlsx")

print("Headers formatted and column widths adjusted!")

**Step 3: Add Formulas**

1. **Calculate the total for each row.**

for row in range(2, len(data) + 2): # Rows 2 to data length

ws[f"E{row}"] = f"=C{row}\*D{row}" # Formula: Quantity \* Price

wb.save("sales\_data.xlsx")

print("Formulas added!")

**Step 4: Apply Conditional Formatting**

1. **Highlight rows where total sales exceed $50.**

from openpyxl.formatting.rule import CellIsRule

from openpyxl.styles import PatternFill

# Apply conditional formatting

fill = PatternFill(start\_color="FFC7CE", end\_color="FFC7CE", fill\_type="solid")

rule = CellIsRule(operator="greaterThan", formula=["50"], fill=fill)

ws.conditional\_formatting.add("E2:E100", rule)

wb.save("sales\_data.xlsx")

print("Conditional formatting applied!")

**Step 5: Extract Data**

1. **Copy rows where "Widget A" was sold into a new worksheet.**

ws\_filtered = wb.create\_sheet("Filtered Data")

# Add headers

ws\_filtered.append(headers)

# Filter data and copy rows

for row in ws.iter\_rows(min\_row=2, max\_row=ws.max\_row, values\_only=True):

if row[1] == "Widget A": # Product column

ws\_filtered.append(row)

wb.save("sales\_data.xlsx")

print("Filtered data extracted!")

**Step 6: Save and Test**

1. Open the file sales\_data.xlsx in Excel and verify:
   * The data is correctly added.
   * Formatting is applied.
   * Formulas and conditional formatting work.
   * Filtered data is in the new worksheet.