Part 5C - Data Validation, Data Cleaning, and Excel Reports Using ChatGPT Scripts

Yahya Nazer

2026.06.18

Table of contents

Part 5C - Data Validation, Data Cleaning, and Excel Reports Using ChatGPT Scripts

Purpose of This Part

In Part 5A and Part 5B, you learned how to read CSV files, write CSV files, read Excel files, write Excel files, and build simple Excel GUI tools.

In Part 5C, you will learn how to ask ChatGPT to generate Python programs that validate, clean, combine, and report on Excel data.

This is one of the most useful business skills in Python because many real-world Excel files contain problems.

Examples:

- Missing values
- Duplicate rows
- Extra spaces
- Wrong dates
- Wrong numbers
- Inconsistent column names
- Multiple files that need to be combined

The goal is to learn this workflow:

```
Business Problem
        ↓
ChatGPT Script
        ↓
Generated Python Program
        ↓
Test
        ↓
Improve Script
        ↓
Improve Python Program
```

Learning Objectives

After completing Part 5C, you will be able to:

- ☐ Write ChatGPT scripts for Excel validation
- ☐ Detect missing values

- ☐ Detect duplicate rows
- ☐ Detect blank rows
- ☐ Clean text columns
- ☐ Remove duplicates
- ☐ Combine multiple Excel files
- ☐ Create Excel reports
- ☐ Save logs and summary files
- ☐ Ask ChatGPT to improve data programs

Course Coding Standard

Every Python program should include:

- Program Name
- Purpose
- User ChatGPT Script
- Expected Output
- Version
- Clearly labeled steps
- Beginner-friendly comments

Example:

```
# ============================================================
# STEP 1 - Import Libraries
# ============================================================
```

Standard Folder Structure

Use this folder structure:

```
ChatGPT-Python-Course
   A-Data
   B-Engine
   C-Results
   D-Documentation
```

Chapter 9 - Data Validation

Objective

Create a Python program that checks an Excel file for common data problems.

The program should find:

- Missing values
- Duplicate rows
- Blank rows
- Column names
- Row count
- Column count

Python File

Create:

```
B-Engine/Validate-Excel.py
```

ChatGPT Script

Copy this into ChatGPT:

```
Write a Python program.

Project folder structure:
Top Folder
+- A-Data
+- B-Engine
+- C-Results
+- D-Documentation

Program Name:
Validate-Excel.py

Program purpose:
Read an Excel file named A-Data/Sales.xlsx and create a validation report.

Requirements:
1. Use pandas.
2. Read A-Data/Sales.xlsx.
3. Count rows.
4. Count columns.
5. List column names.
6. Count missing values by column.
7. Count duplicate rows.
8. Count completely blank rows.
9. Save validation report to C-Results/Validation-Report.txt.
10. Print summary to screen.

Coding Standards:
1. Add a detailed program header at the top of the Python file.
2. The header must include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide the code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make the code suitable for beginners.
6. Use UTF-8 when writing text files.

Expected output:
A text validation report is saved to C-Results.
```

ChatGPT Generated Python Code

```
# ============================================================
# Program Name:
# Validate-Excel.py
#
# Purpose:
# Read an Excel file and create a validation report.
#
# User ChatGPT Script:
# Write a Python program that reads A-Data/Sales.xlsx,
# counts rows and columns, lists column names, checks missing
# values, duplicate rows, blank rows, and saves a report.
#
# Expected Output:
# A text validation report is saved to C-Results.
#
# Version:
# 1.0
# ============================================================

# ============================================================
# STEP 1 - Import Libraries
# ============================================================

# Import Path to work with files and folders.
from pathlib import Path

# Import pandas to read and analyze Excel files.
import pandas as pd

# ============================================================
# STEP 2 - Define Input and Output Files
# ============================================================

# Define the input Excel file.
input_file = Path("A-Data") / "Sales.xlsx"

# Define the results folder.
results_folder = Path("C-Results")

# Create the results folder if it does not exist.
results_folder.mkdir(exist_ok=True)

# Define the output report file.
report_file = results_folder / "Validation-Report.txt"

# ============================================================
# STEP 3 - Read Excel File
# ============================================================

# Read the Excel file into a pandas DataFrame.
data = pd.read_excel(input_file)
```

```
# ============================================================
# STEP 4 - Calculate Validation Information
# ============================================================

# Count rows.
row_count = len(data)

# Count columns.
column_count = len(data.columns)

# Get column names.
column_names = list(data.columns)

# Count missing values by column.
missing_values = data.isna().sum()

# Count duplicate rows.
duplicate_count = data.duplicated().sum()

# Count completely blank rows.
blank_row_count = data.isna().all(axis=1).sum()

# ============================================================
# STEP 5 - Build Report Text
# ============================================================

# Create a list to store report lines.
report_lines = []

# Add report title.
report_lines.append("Excel Validation Report")
report_lines.append("=======================")
report_lines.append("")

# Add row and column counts.
report_lines.append(f"Rows: {row_count}")
report_lines.append(f"Columns: {column_count}")
report_lines.append("")

# Add column names.
report_lines.append("Column Names:")
for column in column_names:
    report_lines.append(f"- {column}")

report_lines.append("")

# Add missing values.
report_lines.append("Missing Values by Column:")
report_lines.append(str(missing_values))
report_lines.append("")

# Add duplicate and blank row counts.
report_lines.append(f"Duplicate Rows: {duplicate_count}")
```

```
report_lines.append(f"Completely Blank Rows: {blank_row_count}")

# Join lines into one text string.
report_text = "\n".join(report_lines)

# ============================================================
# STEP 6 - Save and Print Report
# ============================================================

# Save the report using UTF-8.
report_file.write_text(report_text, encoding="utf-8")

# Print the report.
print(report_text)

# Print saved location.
print()
print("Validation report saved to:", report_file)

# ============================================================
# STEP 7 - End Program
# ============================================================

# Print completion message.
print("Program finished.")
```

Run the Program

```
python Validate-Excel.py
```

On some Mac computers:

```
python3 Validate-Excel.py
```

Improve the Script

Ask ChatGPT:

```
Please improve the program.

New requirements:
1. Save missing value counts to Excel.
2. Save duplicate rows to Excel.
3. Save blank rows to Excel.
4. Create one workbook named Data-Validation-Details.xlsx.
5. Use separate worksheets for Summary, Missing Values, Duplicates, and Blank Rows.
6. Keep the program header and step comments.
```

Exercise

Write a ChatGPT script for a program that validates a customer Excel file and checks:

- Missing email
- Missing phone number
- Duplicate customer names
- Blank rows

What You Learned

You learned how to ask ChatGPT to create Excel data validation programs.

Chapter 10 - Data Cleaning

Objective

Create a Python program that cleans an Excel file.

The program should:

- Remove blank rows
- Remove duplicate rows
- Trim extra spaces from text columns
- Standardize column names
- Save a cleaned Excel file

Python File

Create:

```
B-Engine/Clean-Excel.py
```

ChatGPT Script

Copy this into ChatGPT:

```
Write a Python program.

Project folder structure:
Top Folder
+- A-Data
+- B-Engine
+- C-Results
+- D-Documentation

Program Name:
Clean-Excel.py

Program purpose:
Read A-Data/Sales.xlsx, clean the data, and save a cleaned Excel file.
```

```
Requirements:
1. Use pandas.
2. Read A-Data/Sales.xlsx.
3. Remove completely blank rows.
4. Remove duplicate rows.
5. Standardize column names:
   - strip spaces
   - convert to lowercase
   - replace spaces with underscores
6. Trim extra spaces from all text columns.
7. Save cleaned data to C-Results/Sales-Cleaned.xlsx.
8. Print original row count and cleaned row count.

Coding Standards:
1. Add a detailed program header.
2. Include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make code beginner friendly.

Expected output:
A cleaned Excel file is saved to C-Results.
```

ChatGPT Generated Python Code

```
# ============================================================
# Program Name:
# Clean-Excel.py
#
# Purpose:
# Read Sales.xlsx, clean the data, and save a cleaned Excel file.
#
# User ChatGPT Script:
# Read A-Data/Sales.xlsx, remove blank rows and duplicates,
# standardize column names, trim spaces, and save cleaned file.
#
# Expected Output:
# A cleaned Excel file is saved to C-Results.
#
# Version:
# 1.0
# ============================================================

# ============================================================
# STEP 1 - Import Libraries
# ============================================================

from pathlib import Path
import pandas as pd

# ============================================================
```

```
# STEP 2 - Define Files and Folders
# ============================================================

input_file = Path("A-Data") / "Sales.xlsx"
results_folder = Path("C-Results")
results_folder.mkdir(exist_ok=True)
output_file = results_folder / "Sales-Cleaned.xlsx"

# ============================================================
# STEP 3 - Read Excel File
# ============================================================

data = pd.read_excel(input_file)

# Save original row count for reporting.
original_rows = len(data)

# ============================================================
# STEP 4 - Remove Blank Rows
# ============================================================

# Remove rows where every cell is blank.
data = data.dropna(how="all")

# ============================================================
# STEP 5 - Remove Duplicate Rows
# ============================================================

# Remove completely duplicate rows.
data = data.drop_duplicates()

# ============================================================
# STEP 6 - Standardize Column Names
# ============================================================

# Clean column names by removing spaces and using lowercase.
data.columns = (
    data.columns
    .str.strip()
    .str.lower()
    .str.replace(" ", "_")
)

# ============================================================
# STEP 7 - Clean Text Columns
# ============================================================

# Find text columns.
text_columns = data.select_dtypes(include=["object"]).columns

# Trim spaces from text columns.
for column in text_columns:
    data[column] = data[column].astype(str).str.strip()
```

```
# ============================================================
# STEP 8 - Save Cleaned File
# ============================================================

data.to_excel(output_file, index=False)

# ============================================================
# STEP 9 - Display Summary
# ============================================================

cleaned_rows = len(data)

print("Original rows:", original_rows)
print("Cleaned rows :", cleaned_rows)
print("Rows removed :", original_rows - cleaned_rows)
print("Cleaned file saved to:", output_file)

# ============================================================
# STEP 10 - End Program
# ============================================================

print("Program finished.")
```

Improve the Script

Ask ChatGPT:

```
Please improve the program.

New requirements:
1. Save removed duplicate rows to C-Results/Duplicate-Rows.xlsx.
2. Save removed blank rows to C-Results/Blank-Rows.xlsx.
3. Create a cleaning log text file.
4. Add timestamp to output filenames.
5. Keep the program header and step comments.
```

Exercise

Write a ChatGPT script to clean a customer file:

- Standardize email to lowercase
- Trim customer names
- Remove duplicate emails
- Save a cleaned customer file

What You Learned

You learned how to ask ChatGPT to generate Excel cleaning programs.

Chapter 11 - Combining Excel Files

Objective

Create a Python program that combines multiple Excel files from a folder into one master Excel file.

This is useful when you receive monthly, weekly, or departmental reports.

Folder Setup

Place Excel files in:

```
A-Data/Excel-Files
```

Example:

```
Sales-January.xlsx
Sales-February.xlsx
Sales-March.xlsx
```

Python File

Create:

```
B-Engine/Combine-Excel-Files.py
```

ChatGPT Script

Copy this into ChatGPT:

```
Write a Python program.

Project folder structure:
Top Folder
+- A-Data
+- B-Engine
+- C-Results
+- D-Documentation

Program Name:
Combine-Excel-Files.py

Program purpose:
Combine all Excel files from A-Data/Excel-Files into one master Excel file.

Requirements:
1. Use pathlib.
2. Use pandas.
3. Read all .xlsx files from A-Data/Excel-Files.
4. Skip files whose names start with "~$".
5. Add a column named source_file with the original filename.
6. Combine all rows into one DataFrame.
7. Save the master file as C-Results/Master-Excel-File.xlsx.
```

```
8. Print how many files were processed.
9. Print how many total rows were combined.
10. If no files are found, show a friendly message.

Coding Standards:
1. Add a detailed program header.
2. Include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make code beginner friendly.

Expected output:
A master Excel file is saved to C-Results.
```

ChatGPT Generated Python Code

```python
# ============================================================
# Program Name:
# Combine-Excel-Files.py
#
# Purpose:
# Combine all Excel files from A-Data/Excel-Files into one master Excel file.
#
# User ChatGPT Script:
# Read all .xlsx files from A-Data/Excel-Files, skip temporary files,
# add source_file column, combine rows, and save master workbook.
#
# Expected Output:
# A master Excel file is saved to C-Results.
#
# Version:
# 1.0
# ============================================================

# ============================================================
# STEP 1 - Import Libraries
# ============================================================

from pathlib import Path
import pandas as pd

# ============================================================
# STEP 2 - Define Folders
# ============================================================

input_folder = Path("A-Data") / "Excel-Files"
results_folder = Path("C-Results")
results_folder.mkdir(exist_ok=True)
output_file = results_folder / "Master-Excel-File.xlsx"

# ============================================================
# STEP 3 - Find Excel Files
```

```
# ============================================================

excel_files = []

for file_path in input_folder.glob("*.xlsx"):

    if file_path.name.startswith("~$"):
        continue

    excel_files.append(file_path)

# ============================================================
# STEP 4 - Check if Files Exist
# ============================================================

if not excel_files:

    print("No Excel files were found in:", input_folder)

else:

    # ============================================================
    # STEP 5 - Read and Combine Files
    # ============================================================

    all_data = []

    for file_path in excel_files:

        print("Reading:", file_path.name)

        data = pd.read_excel(file_path)

        data["source_file"] = file_path.name

        all_data.append(data)

    combined_data = pd.concat(all_data, ignore_index=True)

    # ============================================================
    # STEP 6 - Save Master File
    # ============================================================

    combined_data.to_excel(output_file, index=False)

    # ============================================================
    # STEP 7 - Display Summary
    # ============================================================

    print()
    print("Files processed:", len(excel_files))
    print("Total rows:", len(combined_data))
    print("Master file saved to:", output_file)
```

```
# ============================================================
# STEP 8 - End Program
# ============================================================

print("Program finished.")
```

Improve the Script

Ask ChatGPT:

```
Please improve the program.

New requirements:
1. Use a GUI folder selector to select the folder of Excel files.
2. Ask the user where to save the master file.
3. Add a log window.
4. Save a log file.
5. Keep the program header and step comments.
```

Exercise

Write a ChatGPT script that combines only files containing the word Sales in the filename.

What You Learned

You learned how to ask ChatGPT to generate programs that combine many Excel files.

Chapter 12 - Excel Reports

Objective

Create a Python program that reads an Excel file and generates a report workbook.

The report workbook should include:

- Original data
- Summary worksheet
- Missing values worksheet

Python File

Create:

```
B-Engine/Excel-Report.py
```

ChatGPT Script

Copy this into ChatGPT:

```
Write a Python program.

Project folder structure:
Top Folder
+- A-Data
+- B-Engine
+- C-Results
+- D-Documentation

Program Name:
Excel-Report.py

Program purpose:
Read A-Data/Sales.xlsx and generate an Excel report workbook.

Requirements:
1. Use pandas.
2. Read A-Data/Sales.xlsx.
3. Create a Summary worksheet with:
   - row count
   - column count
   - duplicate row count
4. Create a Missing Values worksheet.
5. Include the original data in a worksheet named Data.
6. Save the report as C-Results/Excel-Report.xlsx.
7. Print the saved file path.

Coding Standards:
1. Add a detailed program header.
2. Include Program Name, Purpose, User ChatGPT Script, Expected Output, and Version.
3. Divide code into clearly labeled steps.
4. Add comments explaining every important line.
5. Make code beginner friendly.

Expected output:
An Excel report workbook is saved to C-Results.
```

ChatGPT Generated Python Code

```
# ============================================================
# Program Name:
# Excel-Report.py
#
# Purpose:
# Read Sales.xlsx and generate an Excel report workbook.
#
# User ChatGPT Script:
# Read A-Data/Sales.xlsx, create Summary, Missing Values,
# and Data worksheets, and save Excel-Report.xlsx.
```

```
#
# Expected Output:
# An Excel report workbook is saved to C-Results.
#
# Version:
# 1.0
# ============================================================

# ============================================================
# STEP 1 - Import Libraries
# ============================================================

from pathlib import Path
import pandas as pd

# ============================================================
# STEP 2 - Define Files
# ============================================================

input_file = Path("A-Data") / "Sales.xlsx"
results_folder = Path("C-Results")
results_folder.mkdir(exist_ok=True)
output_file = results_folder / "Excel-Report.xlsx"

# ============================================================
# STEP 3 - Read Excel File
# ============================================================

data = pd.read_excel(input_file)

# ============================================================
# STEP 4 - Create Summary Data
# ============================================================

summary_data = pd.DataFrame({
    "item": [
        "row_count",
        "column_count",
        "duplicate_row_count"
    ],
    "value": [
        len(data),
        len(data.columns),
        data.duplicated().sum()
    ]
})

# ============================================================
# STEP 5 - Create Missing Values Report
# ============================================================

missing_values = pd.DataFrame({
    "column": data.columns,
```

```
    "missing_count": data.isna().sum().values
})

# ============================================================
# STEP 6 - Save Excel Report
# ============================================================

with pd.ExcelWriter(output_file, engine="openpyxl") as writer:

    data.to_excel(writer, sheet_name="Data", index=False)

    summary_data.to_excel(writer, sheet_name="Summary", index=False)

    missing_values.to_excel(writer, sheet_name="Missing Values", index=False)

# ============================================================
# STEP 7 - Display Result
# ============================================================

print("Excel report saved to:", output_file)

# ============================================================
# STEP 8 - End Program
# ============================================================

print("Program finished.")
```

Improve the Script

Ask ChatGPT:

```
Please improve the program.

New requirements:
1. Add a worksheet named Numeric Summary.
2. Add a worksheet named Column Names.
3. Auto-adjust column widths.
4. Add timestamp to the output filename.
5. Keep the program header and step comments.
```

Exercise

Write a ChatGPT script for a report that summarizes sales by product and region.

What You Learned

You learned how to ask ChatGPT to create Excel report workbooks.

Part 5C Review

Skills Checklist

By the end of Part 5C, you should be able to:

- ☐ Validate Excel files
- ☐ Count missing values
- ☐ Count duplicate rows
- ☐ Clean Excel data
- ☐ Remove blank rows
- ☐ Remove duplicate rows
- ☐ Standardize column names
- ☐ Combine Excel files
- ☐ Create Excel report workbooks
- ☐ Improve ChatGPT scripts for data tasks

Summary

In Part 5C, you learned how to use ChatGPT scripts to generate practical Excel data programs.

You learned that Excel automation is not just about reading and writing files.

It also includes:

```
Validation
Cleaning
Combining
Reporting
Logging
Improving
```

These skills are valuable in business, education, research, finance, quality, and operations.

Looking Ahead

In Part 5D, you will build larger practical Excel projects, including:

- Photo inventory workbook
- Excel analyzer GUI
- Student grade analyzer
- Debugging Excel programs
- Part 5 final review