ChatGPT

PRD: Automating Excel and Presentation Workflows with Python and AI

Introduction

Spreadsheets remain the lingua franca of business. Even as organizations adopt modern data platforms, business stakeholders often demand their data in Excel because it is familiar and flexible. Unfortunately, manual spreadsheet manipulation is slow, error-prone and does not scale. Large analytics teams spend considerable time exporting results from Python or SQL into Excel, adding formulas by hand, and manually copying charts into slides. To deliver reliable insights at scale, teams need to automate these steps and integrate them into robust data pipelines.

This product requirement document (PRD) outlines a solution for automating Excel workflows with Python and leveraging AI tools. It also summarizes how successful analytics teams process data and identifies best practices that this product should support.

How Big Analytics Teams Process Data

Modern analytics projects follow a multi-stage lifecycle. A 2025 article from Prophecy explains that successful projects begin by clearly defining the business problem and identifying relevant data sources. The lifecycle then proceeds through **five stages**: **data ingestion**, **data transformation**, **data analysis and modeling**, **insight generation** and **delivery**【464180270019344†L118-L286】. Ingestion collects raw data from operational databases, APIs and event streams; transformation cleans inconsistent values, standardizes data types and joins disparate datasets [1]; analysis and modeling apply statistical tests or machine-learning algorithms [2]; insight generation interprets results and verifies that findings make sense [3]; and delivery packages insights into dashboards, reports or alerts that fit existing workflows [4].

Across these stages the biggest bottleneck is the **collaboration gap** between business and technical teams [5]. Misunderstandings during transformation can lead to incorrect logic, while lack of context during delivery produces reports that do not drive decisions [6]. The product should therefore encourage early stakeholder engagement and make it easy for analysts to translate business rules into automated spreadsheets.

Data Pipeline Best Practices

Research by Ascend.io suggests several best practices that data leaders adopt to improve pipeline reliability and productivity [7]. These include:

- **Adopt a data product mindset.** Data pipelines should be designed to deliver value to end users rather than focusing solely on technical implementation [8]. Product-minded data engineers ask what insights users need and design reports and dashboards accordingly.

- **Prioritize data integrity.** Quality checks should be embedded at every stage of the pipeline instead of verifying validity only at the end [9].
- **Embrace constant change.** Business logic and data sources evolve; pipelines must be flexible enough to accommodate new requirements without costly rewrites [10].
- **Plan for non-linear scalability and maintainability.** Applying DataOps principles and automating routine work allow teams to build more pipelines without proportional increases in headcount [11].

Automating Excel with Python

Python is the de facto language for data science and can automate many spreadsheet tasks. A talk at PyData Global 2025 notes that Python-based data teams often struggle with the "last mile" of delivering results in Excel; the session explores practical ways to bridge this gap using open-source libraries such as **pandas**, **openpyxl** and **xlwings** [12]. Python allows you to generate templated reports, batch-process Excel files and automate dashboards without writing VBA [13].

The `openpyxl` library is particularly useful for manipulating Excel workbooks. A guide from Analytics Vidhya demonstrates how to compute metrics programmatically: instead of calculating percentages manually, you can insert formulas into cells using the library [14]. For example, the guide computes a **Score %** column by setting each cell's value to an average formula:

```
for row in wb['Sheet1']['B3':'H11']:
    row[6].value = f'=AVERAGE({row[0].coordinate}:{row[3].coordinate})'
```

This loop writes an `AVERAGE` formula for each row, automatically referencing the correct range [14]. Similarly, the Real Python tutorial shows how to add formulas such as `=AVERAGE(H2:H100)` or `=COUNTIF(I2:I100, " >0")` by simply assigning a formula string to a cell [15]. With `openpyxl` you can also merge cells, apply filters, insert charts or images and style your workbook [16].

Python in Excel

In 2024 Microsoft introduced **Python in Excel**, which lets users write Python code directly in spreadsheet cells. According to Microsoft, you can clean, explore and analyze data using popular libraries like pandas, Matplotlib and scikit-learn without leaving Excel [17]. Python formulas can be recalculated automatically and shared securely across the organization [18]. This integration removes the need to export data into separate notebooks and simplifies collaboration [19].

AI Features in Excel

Microsoft's Copilot in Excel uses generative AI to assist with repetitive tasks. It can **clean data** by detecting inconsistencies or duplicates [20], **identify trends** and generate charts that highlight patterns [21], **suggest complex formulas** based on your data [22] and provide **context-aware recommendations** while you work [23]. Copilot can even generate automated reports with summaries and visualizations [24]. These AI features increase efficiency, improve accuracy and support decision-making [25]. Integrating Copilot into automated workflows could further streamline report generation.

Streamlining PowerPoint with Python

Analysts often need to present findings in slides. The `python-pptx` library allows you to programmatically create PowerPoint presentations. Combined with `openpyxl` or pandas, you can read data, generate charts with Matplotlib, and insert them into slides. For example, `python-pptx` can create a bar chart slide summarizing metrics for each customer or product. Though not covered in the cited resources, this workflow complements the Excel automation discussed above.

Product Vision

Problem Statement

Business users demand Excel spreadsheets and slide decks, but manual workflows are error-prone and slow. Analysts need to automate repetitive spreadsheet tasks, ensure data quality and generate polished presentations without leaving their Python ecosystem.

Objectives

1. **Automate spreadsheet preparation.** Read data from diverse sources, clean it and load it into Excel. Insert formulas programmatically to compute averages, percentages and other metrics.
2. **Support the analytics lifecycle.** Enable ingestion, transformation, analysis and delivery within one pipeline【464180270019344†L118-L286】. Emphasize a data-product mindset [8] and integrate data quality checks throughout [9].
3. **Leverage AI tools.** Incorporate Copilot in Excel for data cleaning, trend identification and formula suggestions [26]. Use Python in Excel for advanced analytics and predictive modeling [18].
4. **Generate presentations automatically.** Use Python to build PowerPoint decks with charts and tables directly from Excel data.
5. **Ensure maintainability and scalability.** Provide automated tests and adopt DataOps practices so that adding new reports does not require linear increases in headcount [11].

Key Features

1. **Data ingestion module** – connectors to CSV files, databases and APIs, with configuration defined outside of code. Input data is stored in a common format for downstream processing.
2. **Excel automation module** – functions to create workbooks, write data, insert formulas (e.g. `AVERAGE` formulas per row), apply styles, merge cells, add charts and images [16] [15].
3. **Analytics module** – optionally run statistical analyses or machine-learning models using pandas, scikit-learn or Python-in-Excel; results feed back into spreadsheets and presentations.
4. **AI integration** – wrappers that call Copilot in Excel (when available) for data cleaning, trend identification and formula suggestions [26]. Provide a fallback to built-in logic when AI is unavailable.
5. **Report generation module** – utilities using `python-pptx` to create slides summarizing key metrics. Support templated PowerPoint layouts and automatic chart insertion.
6. **Testing and CI/CD** – include unit tests (e.g. verifying that formulas are inserted correctly) and a CI pipeline to run tests on every change. Adopt DataOps principles to ensure non-linear scalability and maintainability [11].

Proposed Repository Structure

The accompanying repository skeleton demonstrates how to automate Excel tasks with Python and includes a test suite. A compressed archive of the repository is provided separately. The structure is as follows:

Path	Description
`README.md`	Overview of the project, motivation and setup instructions.
`requirements.txt`	Lists Python dependencies (`openpyxl`, `pytest`, `python-pptx`, `pandas`).
`scripts/excel_operations.py`	Library functions for reading CSV files, writing Excel workbooks and inserting `AVERAGE` formulas per row [14]. Includes a `main()` function for command-line use.
`scripts/generate_report.py`	Placeholder for future report or PowerPoint generation logic.
`data/sample_input.csv`	Sample tabular data used in the example.
`tests/test_excel_operations.py`	`pytest` tests verifying that formulas are inserted correctly and that the header includes an “Average” column.

By running `python -m scripts.excel_operations`, users can generate an Excel workbook from the sample data with working average formulas. The tests can be executed with `pytest` to ensure correct functionality.

Conclusion

Automating Excel and presentation workflows requires a combination of robust data engineering practices and modern scripting languages. Successful analytics teams treat data as a product and orchestrate ingestion, transformation, analysis, insight generation and delivery【464180270019344†L118-L286】. They embed data quality checks and plan for change and scalability [27]. Python’s open-source ecosystem (e.g. `openpyxl`, pandas, python-pptx) enables repeatable, testable spreadsheet automation [14] [15], while AI features in Excel such as Copilot accelerate data cleaning, formula creation and insight generation [26]. The repository provided with this PRD offers a foundation for implementing these ideas, empowering analytics teams to work more efficiently and deliver trustworthy insights.

---

1 2 3 4 5 6 The End-to-End Analytics Lifecycle: From Ingestion to Insight | Prophecy
https://www.prophecy.io/blog/the-end-to-end-analytics-lifecycle-from-ingestion-to-insight

7 8 9 10 11 27 Five Data Pipeline Best Practices to Follow in 2025
https://www.ascend.io/blog/data-pipeline-best-practices

12 13 Python Meets Excel: Smarter Workflows for Analysts and Data Teams :: PyData Global 2025 :: pretalx
https://cfp.pydata.org/pydataglobal2025/talk/HCURNN/

14 16 Master Guide for Excel Automation Using Python - Analytics Vidhya
https://www.analyticsvidhya.com/blog/2023/05/master-guide-for-excel-automation-using-python/

15 A Guide to Excel Spreadsheets in Python With openpyxl – Real Python
https://realpython.com/openpyxl-excel-spreadsheets-python/

17 18 19 Using Python in Excel for Data Analysis | Microsoft 365
https://www.microsoft.com/en-us/microsoft-365/python-in-excel

20 21 22 23 24 25 26 AI in Excel – Features and Benefits | Microsoft Excel
https://www.microsoft.com/en-us/microsoft-365/excel/ai-for-excel