PharmaSUG 2025 - Paper SD - 257

Enhancing Clarity and Efficiency in Clinical Statistical Programming Task Management:

An Automated Integrated Task Reporting Solution with Excel and Python

Xinran Hu, Merck & Co., Inc., Rahway, NJ, USA

Abstract

Effective task management and timely communication are critical in clinical statistical programming, where multiple studies, tight timelines, and cross-functional collaboration converge. This paper presents an integrated Python-based solution to automate task reporting across weekly, monthly, quarterly, and yearly intervals. By extracting data from an Excel-based task tracking tool, the system categorizes tasks based on due dates (past week, current week, upcoming week, and quarterly milestones) and priority levels.

Personalized emails are automatically generated and scheduled for delivery to team members and leaders at predefined intervals—such as every Monday at 9 AM, the first day of each month/quarter/year, or at the end of a reporting period for summary analysis. For team leaders, the system provides a consolidated report highlighting high-priority milestones along with an analytical summary of task completion rates, categorized by team members, study, project, or compound.

Built with Python, the solution leverages pandas for efficient data handling and pywin32 for seamless Outlook integration, while Windows Task Scheduler ensures fully automated execution without manual intervention. By delivering structured, timely, and actionable reports, this system significantly reduces task omissions, improves milestone tracking, fosters cross-team alignment, and provides valuable insights into task completion trends—empowering leaders to enhance resource allocation, planning, and project management.

A case study demonstrates the system's application in a clinical programming environment, highlighting substantial time savings, improved task visibility, and enhanced accountability achieved through this automated reporting approach.

1. Introduction

Clinical statistical programming teams operate in fast-paced environments where they must manage programming deliverables across multiple concurrent studies and timelines. Ensuring accurate tracking of task assignments, progress, and handoffs is vital for successful project execution. However, manual task management methods—typically based on Excel spreadsheets and ad hoc emails—often result in delayed updates, missed deadlines, and a lack of visibility into team performance.

These inefficiencies can directly impact submission timelines, collaboration effectiveness, and quality oversight. In particular, programming leads often to lack real-time insights into milestone progress and resource workload, complicating planning and forecasting efforts.

To address these challenges, we designed and implemented a fully automated reporting solution that combines Excel-based task tracking with Python scripting. This tool enables structured data extraction, categorization of tasks by time window and priority, visualization of progress, and personalized email communication on a recurring schedule. The objective is to provide timely, accurate, and actionable task updates to both team members and leadership with minimal manual effort.

2. System Architecture and Design

The system is built around a core Excel workbook that serves as the task database. Each row represents an individual task and contains metadata such as owner, due date, status, study, project, and priority. Python scripts connect to this workbook, read the data into memory using pandas, and filter or group tasks according to preset rules.

Tasks are first categorized based on due dates, enabling the system to generate views such as 'past week completed', 'current week due', 'next week upcoming', and 'quarterly milestones'. Tasks are further segmented by priority to highlight urgent deliverables.

Progress visualization is performed using matplotlib, which generates Gantt-style charts with color-coded bars representing task ownership and completion percentage. These figures are attached to summary emails for improved interpretability.

The email generation module uses pywin32 to interface with Microsoft Outlook. Personalized messages are generated for each assignee, containing their weekly deliverables and a summary of completed work. A separate summary email is created for programming leads, aggregating key metrics and visual summaries.

Finally, the system is integrated with Windows Task Scheduler to execute these scripts on a recurring basis—weekly, monthly, quarterly, or at other relevant reporting intervals.

3. Implementation Details

Task data are loaded from Excel using *pandas' read_excel* method and validated for structure and completeness. Records with missing due dates, assignees, or status fields are flagged for review.

The email module uses pywin32's COM interface to construct and dispatch messages from Outlook. Templates are rendered dynamically to include variable content such as task names, dates, and user-specific greetings. Attachments (e.g., images or exported summaries) are included as needed.

Task progress is visualized via matplotlib by plotting bars on a timeline with corresponding start and end dates, color-coded by completion status. These Gantt-style plots allow users and leads to quickly assess workload distribution and identify delays or concentration points.

All reporting scripts are executed via Windows Task Scheduler using scheduled .bat files or command-line triggers, ensuring automation without the need for third-party tools or manual intervention.

4. Automated Scheduling and Execution

Execution of reports is fully automated using Windows Task Scheduler. Each reporting job is mapped to a separate script and scheduled independently to allow for flexible timing (e.g., Mondays at 9:00 AM for weekly reports, the 1st day of each month for monthly summaries).

Error handling is incorporated through logging and exception handling to ensure that failures are captured, and notification emails are sent if an issue arises. Logging outputs are stored in local directories for review by technical owners.

This configuration ensures consistency in reporting and reduces the reliance on team members to manually run any tools or check for reminders.

5. Case Study: Application for a Statistical Programming Team

This system was piloted by a programming group supporting three Phase 3 studies, each with separate reporting requirements and timelines. Over a 12-week evaluation period, weekly and monthly reports were automatically generated and delivered to 5 team members and 2 programming leads.

Results from the pilot indicated a significant reduction in administrative effort: manual reporting time was reduced by over 70%. Visibility into weekly priorities improved markedly, and programming leads were able to proactively adjust assignments based on data trends such as missed deadlines or overloads.

The Gantt charts embedded in summary emails proved particularly effective in cross-functional discussions, enabling better resource planning and highlighting risks during study transitions and submission preparation.

6. Challenges and Solutions

Several practical challenges emerged during implementation. First, task data quality varied depending on who maintained the tracking spreadsheet. This was mitigated by applying validation checks and conditional formatting rules in Excel.

Second, the scalability of the solution was tested as the number of tasks grew. Script runtime was optimized using vectorized operations in pandas, and visualization code was modularized to reduce overhead.

Finally, scheduling across multiple time zones and user preferences was handled by centralizing execution on a virtual machine and using time zone-aware scheduling rules.

7. Future Enhancements

Planned improvements include integration with platforms such as Jira or Microsoft Project to enable real-time synchronization of task data.

Additional reporting outputs will be developed for Power BI and Tableau to allow for interactive dashboards and trend exploration by management teams.

A potential enhancement includes machine learning-based models that predict task slippage and suggest proactive adjustments to prevent bottlenecks.

We are also exploring a web-based interface for maintaining task records, enabling multi-user access, changing tracking, and remote configuration of reporting schedules.

8. Conclusion

This paper demonstrates the value of combining Python automation with Excel task tracking to improve operational efficiency in clinical programming teams. By generating timely, structured, and role-specific reports, the system strengthens communication, reduces delays, and provides meaningful insights into execution trends.

With minimal infrastructure requirements, the solution can be deployed broadly across teams and scaled to support larger portfolios. As study complexity and programming demands continue to grow, lightweight automation frameworks like this one offer high impact with modest development effort.

Appendix: Key Python Code Implementation and Explanation

This appendix provides three core Python code snippets illustrating the primary logic of the automated Excel-based task management and reporting tool described in this paper.

Code1: Data Extraction and Task Categorization (pandas)

This section of code demonstrates how task data is imported from Excel and categorized according to due dates and task status.

```
import pandas as pd
from datetime import datetime, timedelta
# Load tasks from Excel into pandas DataFrame
tasks = pd.read_excel("Task_Tracking.xlsx", sheet_name="Tasks")
# Define reporting timeframes
today = datetime.now().date()
last_week = today - timedelta(days=7)
next_week = today + timedelta(days=7)
# Categorize tasks
completed_tasks = tasks[
    (tasks["Status"] == "Completed") &
    (tasks["Completion Date"] >= pd.Timestamp(last_week))]
upcoming_tasks = tasks[
    (tasks["Due Date"] > today) &
    (tasks["Due Date"] <= pd.Timestamp(next_week))]
```

Explanation:
The script loads tasks from an Excel spreadsheet, converts relevant date columns to datetime format, and categorizes tasks into "completed in the past week" and "due in the next week" using logical filtering.

Code2: Automated Email Generation (pywin32 with Outlook)

```
import win32com.client as win32
def send_email(to_email, subject, body, attachment=None):
    outlook = win32.Dispatch('outlook.application')
    mail = outlook.CreateItem(0)
    mail.To = to_email
    mail.Subject = subject
    mail.Body = body
    if attachment:
        mail.Attachments.Add(attachment)
  mail.Send()
```

Explanation:

This script uses Outlook integration to automatically compose and send emails with optional file attachments. The email content can be customized dynamically based on individual tasks and assignees.

Code3: Task Progress Visualization (Gantt Chart with matplotlib)

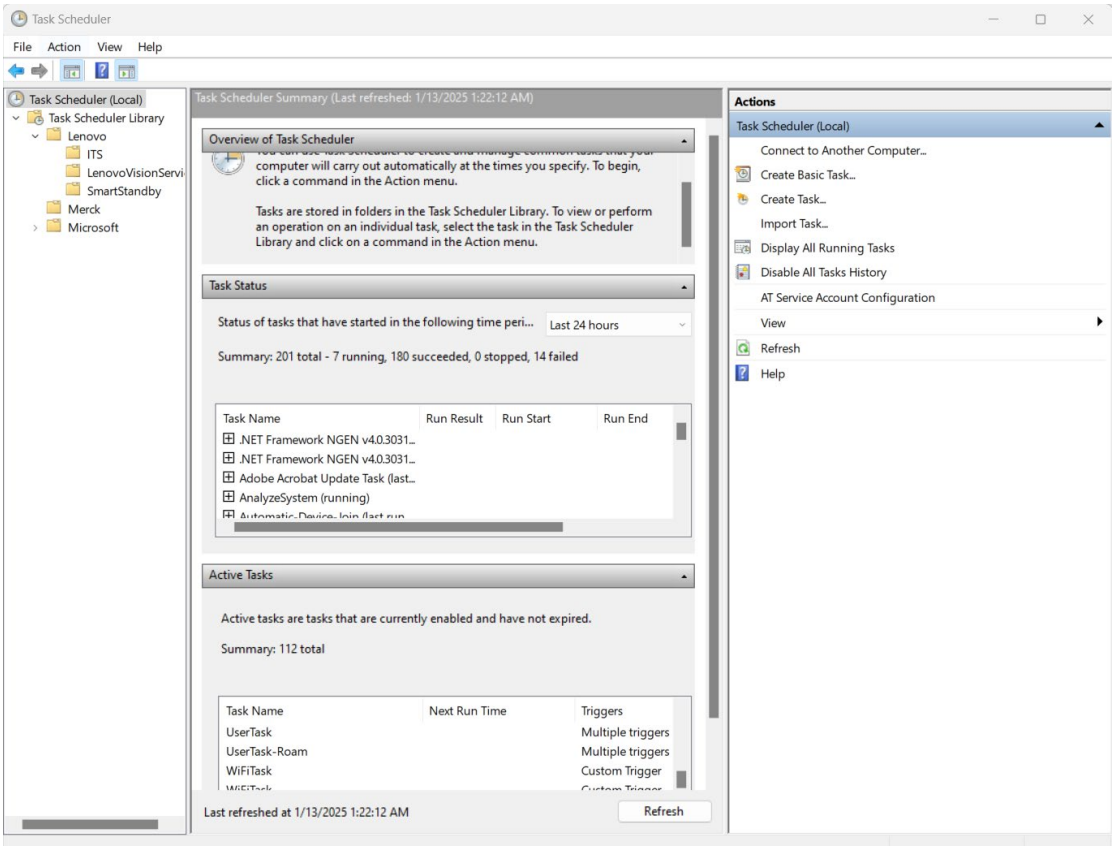
This final snippet demonstrates how the task progress is visually summarized through a Gantt chart.

```
import matplotlib.pyplot as plt
# Example Gantt chart visualization
fig, ax = plt.subplots(figsize=(10, 6))
# Plot tasks
for idx, task in tasks.iterrows():
    ax.barh(task["Task Name"],
            (task["Due Date"] - task["Start Date"]).days,
            left=task["Start Date"],
            color='green' if task["Status"] == 'Completed' else
'skyblue')
```

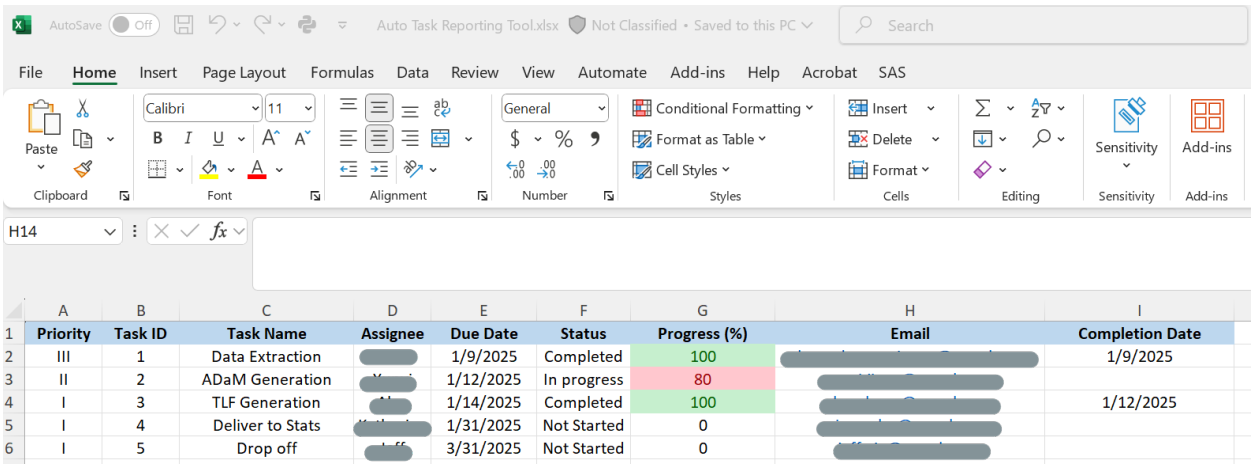
```
ax.set_xlabel('Date')

ax.set_title('Clinical Programming Task Progress')

plt.tight_layout()

plt.show()
```

Explanation:
This visualization clearly displays each task's timeline and completion status through horizontal bars, with colors differentiating completed (green) and ongoing tasks (blue), aiding team members and leadership in quickly assessing overall project status.

Windows Task Scheduler: ensures fully automated execution without manual intervention

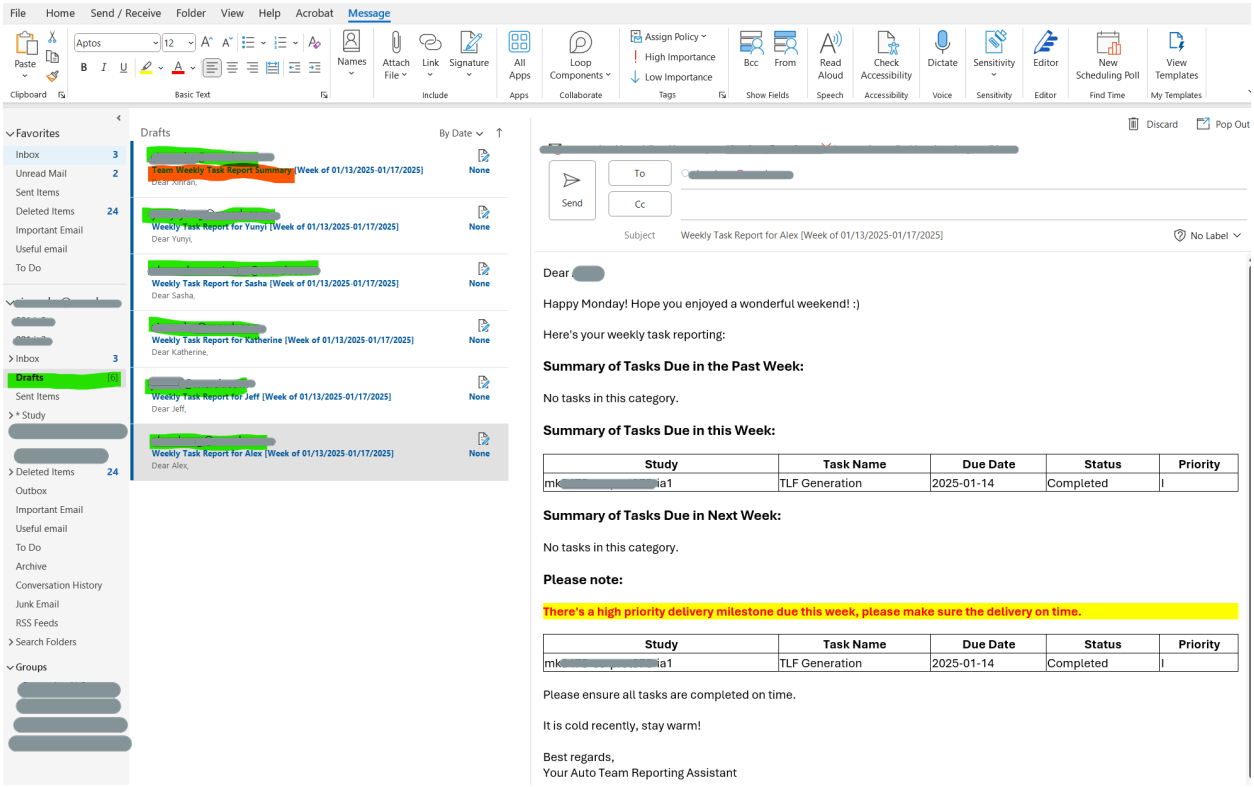


Excel file managing task and timeline:



	A	B	C	D	E	F	G	H	I
1	Priority	Task ID	Task Name	Assignee	Due Date	Status	Progress (%)	Email	Completion Date
2	III	1	Data Extraction	[redacted]	1/9/2025	Completed	100	[redacted]	1/9/2025
3	II	2	ADaM Generation	[redacted]	1/12/2025	In progress	80	[redacted]	
4	I	3	TLF Generation	[redacted]	1/14/2025	Completed	100	[redacted]	1/12/2025
5	I	4	Deliver to Stats	[redacted]	1/31/2025	Not Started	0	[redacted]	
6	I	5	Drop off	[redacted]	3/31/2025	Not Started	0	[redacted]	

Email Drafts Box shown below after program automatically rerunning by task scheduler:



Drafts

Team Weekly Task Report Summary [Week of 01/13/2025-01/17/2025]
Dear Xinran,

Weekly Task Report for Yunyi [Week of 01/13/2025-01/17/2025]
Dear Yunyi,

Weekly Task Report for Sasha [Week of 01/13/2025-01/17/2025]
Dear Sasha,

Weekly Task Report for Katherine [Week of 01/13/2025-01/17/2025]
Dear Katherine,

Weekly Task Report for Jeff [Week of 01/13/2025-01/17/2025]
Dear Jeff,

Weekly Task Report for Alex [Week of 01/13/2025-01/17/2025]
Dear Alex,

Subject: Weekly Task Report for Alex [Week of 01/13/2025-01/17/2025]

Dear [redacted]

Happy Monday! Hope you enjoyed a wonderful weekend! :)

Here's your weekly task reporting:

Summary of Tasks Due in the Past Week:

No tasks in this category.

Summary of Tasks Due in this Week:

Study	Task Name	Due Date	Status	Priority
mk[redacted]ia1	TLF Generation	2025-01-14	Completed	I

Summary of Tasks Due in Next Week:

No tasks in this category.

Please note:

There's a high priority delivery milestone due this week, please make sure the delivery on time.

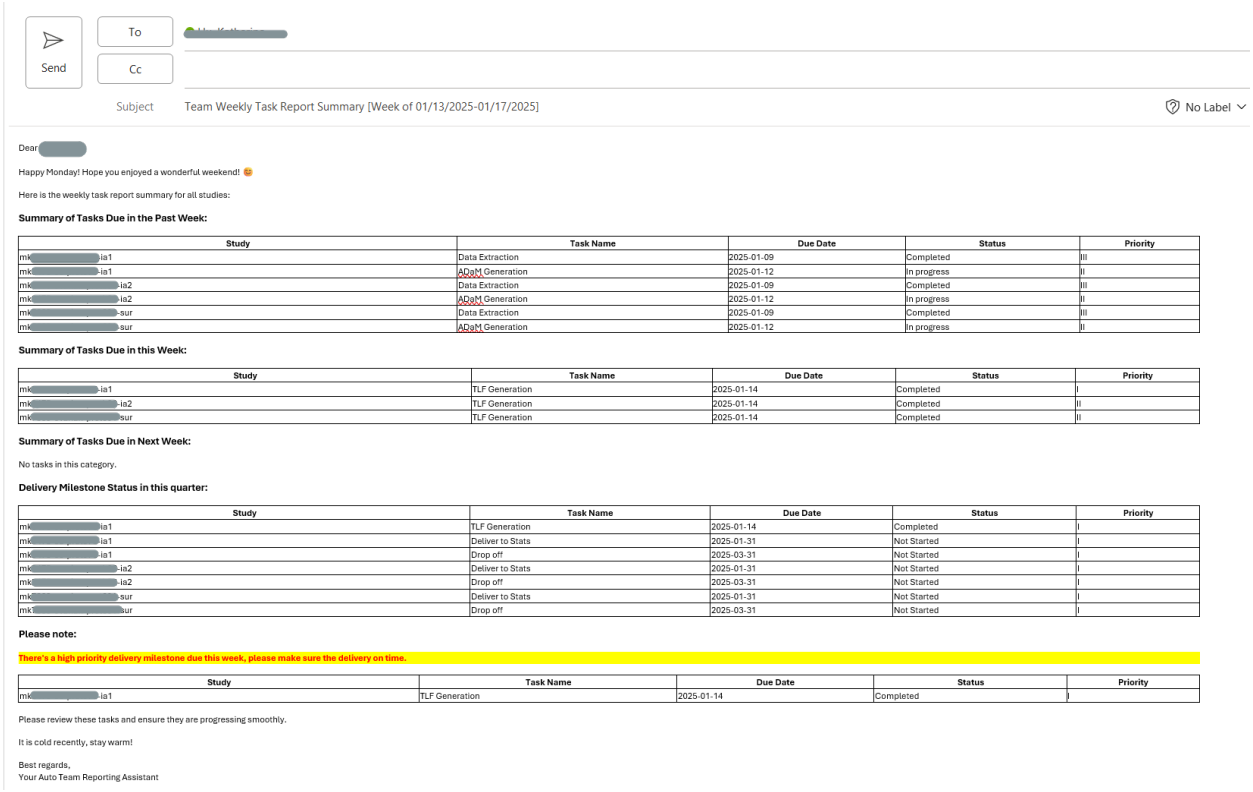
Study	Task Name	Due Date	Status	Priority
mk[redacted]ia1	TLF Generation	2025-01-14	Completed	I

Please ensure all tasks are completed on time.

It is cold recently, stay warm!

Best regards,
Your Auto Team Reporting Assistant

Email Output: Summary report of task across all studies to team/programming lead:



Send | To | Cc

Subject: Team Weekly Task Report Summary [Week of 01/13/2025-01/17/2025] | No Label

Dear [redacted]

Happy Monday! Hope you enjoyed a wonderful weekend!

Here is the weekly task report summary for all studies:

Summary of Tasks Due in the Past Week:

Study	Task Name	Due Date	Status	Priority
mk[redacted]ia1	Data Extraction	2025-01-09	Completed	III
mk[redacted]-ia1	ADaM Generation	2025-01-12	In progress	II
mk[redacted]-ia2	Data Extraction	2025-01-09	Completed	III
mk[redacted]-ia2	ADaM Generation	2025-01-12	In progress	II
mk[redacted]-sur	Data Extraction	2025-01-09	Completed	III
mk[redacted]-sur	ADaM Generation	2025-01-12	In progress	II

Summary of Tasks Due in this Week:

Study	Task Name	Due Date	Status	Priority
mk[redacted]-ia1	TLF Generation	2025-01-14	Completed	I
mk[redacted]-ia2	TLF Generation	2025-01-14	Completed	II
mk[redacted]sur	TLF Generation	2025-01-14	Completed	II

Summary of Tasks Due in Next Week:

No tasks in this category.

Delivery Milestone Status in this quarter:

Study	Task Name	Due Date	Status	Priority
mk[redacted]ia1	TLF Generation	2025-01-14	Completed	I
mk[redacted]ia1	Deliver to Stats	2025-01-31	Not Started	I
mk[redacted]-ia1	Drop off	2025-03-31	Not Started	I
mk[redacted]-ia2	Deliver to Stats	2025-01-31	Not Started	I
mk[redacted]-ia2	Drop off	2025-03-31	Not Started	I
mk[redacted]-sur	Deliver to Stats	2025-01-31	Not Started	I
mk1[redacted]sur	Drop off	2025-03-31	Not Started	I

Please note:

There's a high priority delivery milestone due this week, please make sure the delivery on time.

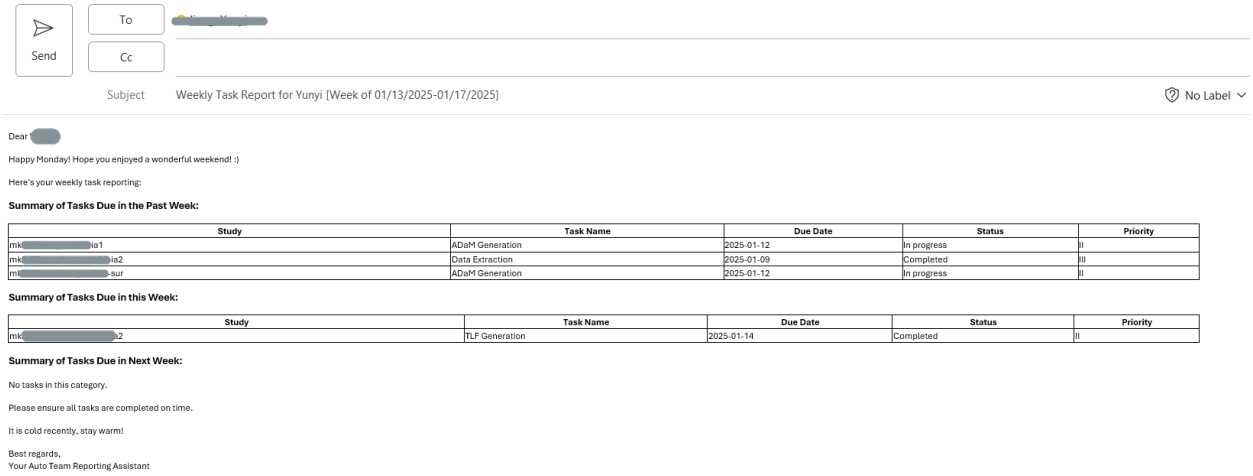
Study	Task Name	Due Date	Status	Priority
mk[redacted]ia1	TLF Generation	2025-01-14	Completed	I

Please review these tasks and ensure they are progressing smoothly.

It is cold recently, stay warm!

Best regards,
Your Auto Team Reporting Assistant

Email output: Customized report to each task assignee:



Send | To | Cc

Subject: Weekly Task Report for Yunyi [Week of 01/13/2025-01/17/2025] | No Label

Dear [redacted]

Happy Monday! Hope you enjoyed a wonderful weekend! :)

Here's your weekly task reporting:

Summary of Tasks Due in the Past Week:

Study	Task Name	Due Date	Status	Priority
mk[redacted]ia1	ADaM Generation	2025-01-12	In progress	II
mk[redacted]ia2	Data Extraction	2025-01-09	Completed	III
mk[redacted]-sur	ADaM Generation	2025-01-12	In progress	II

Summary of Tasks Due in this Week:

Study	Task Name	Due Date	Status	Priority
mk[redacted]a2	TLF Generation	2025-01-14	Completed	II

Summary of Tasks Due in Next Week:

No tasks in this category.

Please ensure all tasks are completed on time.

It is cold recently, stay warm!

Best regards,
Your Auto Team Reporting Assistant

References

1. U.S. Food & Drug Administration. Providing Regulatory Submissions in Electronic Format — Standardized Study Data. Guidance for Industry.
2. Python Software Foundation. https://www.python.org/
3. pandas Development Team. https://pandas.pydata.org/

Acknowledgments

I would like to express my sincere gratitude to Jeff Xia, for his valuable insights, thoughtful suggestions, and careful review throughout the preparation of this paper. His guidance and expertise have greatly enhanced the clarity and quality of this work.

Contact Information

Your comments and questions are valued and encouraged. Contact the author at:

Xinran Hu
Enterprise: Merck & Co., Inc.
Address: 126 East Lincoln Avenue, City, State ZIP: Rahway, NJ 07065-4607
Phone: 732- 5946154
Email: xinran.hu@merck.com
Web: www.merck.com