Business Context:

As a business leader at Orion Tech Solutions, you ("Alex Carter") oversee multiple software development and IT infrastructure projects. Your responsibilities include coordinating with stakeholders, managing escalations, and ensuring timely deliveries. With hundreds of emails flooding your inbox daily, manually sorting through them is time-consuming and increases the risk of missing critical updates, client escalations, or project approvals.

Objective: The goal of this project is to develop a Generative AI-powered system that: ✅ Summarizes emails into actionable insights using the Yesterbox approach (excluding today's emails). ✅ Prioritizes emails based on urgency, sender, and context. ✅ Draft context-aware responses to reduce manual effort. ✅ Evaluate the drafted context-aware responses using LLM-as-a-Judge. Tasks & Workflow Task 1: Generate a Detailed Summary of Yesterday's Inbox Task 1A: Executive Dashboard (Top-Level Summary of Yesterday's Emails) Sample Output: 🔹 Total Emails from Yesterday: 100 🔹 🔴 Urgent & High-Priority Emails: 10 (Require Immediate Action Today) 🔹 ⚡ Deadline-Driven Emails: 8 (Must Be Addressed Today) 🔹 📌 Routine Updates & Check-ins: 35 (Review & Acknowledge) 🔹 ⛄ Non-Urgent & Informational Emails: 45 (Can Be Deferred or Delegated) 🔹 🎉 Personal & Social Emails: 22 (Optional Review) 🔹 🗑️ Spam/Unimportant Emails Filtered Out: 20

AI Conclusion: "You have 18 critical emails from yesterday that require action today. Additionally, there are 35 updates to review at your convenience."

Task 1B: Analyze Urgent & High-Priority Emails (🔴 Must-Do First Today) Focus on emails that require immediate action and impact critical projects or client relationships.

Task 1C: Review Deadline-Driven Emails (⚡ Needs Attention Today) Identify emails tied to important deadlines and ensure timely responses.

Task 2: AI-Generated Response Drafts for Critical Email For each Urgent & High-priority email or Deadline-Driven email from yesterday, generate an AI-powered response draft for quick review and editing before sending.

NOTE : Critical Emailsare the combination of Urgent & High-Priority Emails + Deadline-Driven Emails

Task 3: Validate AI-Generated Results Using the "LLM as a Judge" Technique To ensure accuracy and reliability, apply the "LLM as a Judge" technique to evaluate: ✅ Relevance: How well does thesummaryaddress the input query or task? ✅ Clarity: How clear and understandable is thesummary? ✅ Actionability: Does thesummaryprovide clear next steps or actionable information? ✅ Strengths: Highlight the key strengths of the summary. ✅ Improvements: Suggest 1-2 areas for improvement. ✅ Overall Justification: Provide a 2-3 line summary evaluation, including key observations.

```
import json
import os
from pathlib import Path
import requests

# Define file paths
local_cfg_path = Path('/content/JHU_Learnings/Week 9 Project/config.json')
github_url = "https://raw.githubusercontent.com/linufx2208-sketch/JHU_Learnings/main/Week%209%20Project/config.json"

API_KEY = None

# Ensure the directory exists
os.makedirs(local_cfg_path.parent, exist_ok=True)

# Attempt to download config.json from GitHub if it doesn't exist
if not local_cfg_path.exists():
    print(f'{local_cfg_path} not found. Attempting to download from GitHub...')
    try:
        response = requests.get(github_url)
        response.raise_for_status() # Raise an HTTPError for bad responses (4xx or 5xx)
        with open(local_cfg_path, 'w', encoding='utf-8') as f:
            f.write(response.text)
        print(f"Successfully downloaded {local_cfg_path} from GitHub.")
    except requests.exceptions.RequestException as e:
        print(f"Error fetching config.json from GitHub: {e}. Please ensure the GitHub URL is correct and accessible.")
    except Exception as e:
        print(f"An unexpected error occurred during download: {e}.")

# Now, attempt to load API_KEY from the local `config.json` file
if local_cfg_path.exists():
    try:
        with open(local_cfg_path, 'r', encoding='utf-8') as f:
```

```
            cfg = json.load(f)
        API_KEY = cfg.get('API_KEY') or cfg.get('api_key') or cfg.get('openai_api_key')
    except Exception as e:
        print(f"Error loading API_KEY from {local_cfg_path}: {e}")
        API_KEY = None
else:
    print(f'{local_cfg_path} still not found after download attempt. API_KEY cannot be loaded.')

# Provide a short confirmation but do not print the secret itself
if API_KEY:
    print(f'API_KEY loaded from {local_cfg_path} (length {len(API_KEY)})')
else:
    print(f'API_KEY not found or could not be loaded from {local_cfg_path}.')
```

```
API_KEY loaded from /content/JHU_Learnings/Week 9 Project/config.json (length 67)
```

```
# Print a masked form of the API key so the full secret is not exposed
# If API_KEY is set, show first 4 and last 4 characters separated by '...'; otherwise print None
api_key_val = globals().get('API_KEY')
if api_key_val:
    print(api_key_val[:8] + '...' + api_key_val[-8:])
else:
    print(api_key_val)
```

```
gl-U2Fsd...oxDCQTE2
```

```
!pip install -q openai==1.61.1
```

```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 463.1/463.1 kB 9.1 MB/s eta 0:00:00
```

```
import openai
print(openai.__version__)
```

```
1.61.1
```

Loading the `config.json` file

```
# @title Loading the `config.json` file
import json, os

# Load the JSON file and extract values
file_name = '/content/JHU_Learnings/Week 9 Project/config.json'
with open(file_name, 'r') as file:
    config = json.load(file)
    os.environ['OPENAI_API_KEY']  = config.get("API_KEY") # Loading the API Key
    os.environ["OPENAI_BASE_URL"] = config.get("OPENAI_API_BASE") # Loading the API Base Url
```

```
model_name = "gpt-4o-mini"
```

```
from openai import OpenAI

# Initialize OpenAI client
client = OpenAI()
```

LLM function

```
# @title LLM function
# @markdown Once the API details are filled, the notebook v


def llm(system_prompt, user_prompt):
  try:
      # Craft the messages to pass to chat.completions.crea
      prompt = [
          {'role':'system', 'content': system_prompt},
          {'role': 'user', 'content': user_prompt}
      ]

      response = client.chat.completions.create(
          model=model_name,
```

Once the API details are filled, the notebook will automatically load the configuration, and learners can generate model outputs using the llm() function.

Hide code

```
            messages=prompt,
            temperature=0
        )

        return response.choices[0].message.content.strip()

    except Exception as e:
        prediction = f'Sorry, I encountered the following err
        print(prediction)
```

Step 1: Load the Dataset

```
# @title Step 1: Load the Dataset
# Data Loading

import pandas as pd
# Use the raw GitHub URL for the CSV file
df = pd.read_csv("https://raw.githubusercontent.com/linufx2208-sketch/JHU_Learnings/main/Week%209%20Project/Alex_emails_marc
#df = pd.read_csv("/content/JHU_Learnings/Week 9 Project/Alex_emails_march_04.csv", index_col="email_id", encoding='latin-1'
df
```

Step 2: Apply Yesterbox Filtering

```
# @title Step 2: Apply Yesterbox Filtering
# @markdown The Yesterbox approach involves processing emai

# @markdown For this dataset, consider today's date as 4th

# @markdown We filter the dataset to only include emails re
# (Yesterbox Approach)(Today: 4 march)



from datetime import datetime, timedelta

yesterday_date = pd.to_datetime("3/3/2025").strftime('%m/%d

df['date_received'] = pd.to_datetime(df['date_received']).d

yesterday_emails = df[df['date_received'] == yesterday_date
print(f"Filtered Emails Count: {len(yesterday_emails)}")
```

The Yesterbox approach involves processing emails from the previous day first before tackling today's emails.

For this dataset, consider today's date as 4th March 2025.

We filter the dataset to only include emails received on 3rd March 2025 (yesterday)

Hide code

```
Filtered Emails Count: 51
```

```
df.shape
```

```
(60, 5)
```

```
# Here we see only 51 emails, as in 10 email had the date of 4th March 2025
yesterday_emails.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51 entries, 0 to 50
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   date_received   51 non-null     object
 1   sender          51 non-null     object
 2   subject         51 non-null     object
 3   body            51 non-null     object
 4   main_recipient  51 non-null     object
dtypes: object(5)
memory usage: 2.1+ KB
```

TASK - Categorization of emails

Your task is to write the `system_prompt` & `user_prompt` for classifying all the emails into one of the below pre-defined categories.

1. **Urgent & High-Priority Emails**:
    - Emails that require immediate action and must be addressed today.

2. **Deadline-Driven Emails**:
    - Time-sensitive emails or meeting requests that need attention today.
3. **Routine Updates & Check-ins**:
    - Emails that require review and acknowledgment without immediate action.
4. **Non-Urgent Informational Emails**:
    - Emails that can be deferred or delegated to another time or person.
5. **Personal & Social Emails**:
    - Emails that can be reviewed optionally at a later time.
6. **Spam/Unimportant Emails**:
    - Emails that are not relevant and should be filtered out.

Note: Your response should include only one of the above six specified categories and nothing else.

```
Example Email : 'You Won a Free iPhone 15 Pro! ?? Click to Claim'
Expected Response : 'Spam/Unimportant Emails'
```

Code to add categories to the dataset

```
# @title Code to add categories to the dataset
categories = ['Urgent & High-Priority Emails',
              'Deadline-Driven Emails',
              'Routine Updates & Check-ins',
              'Non-Urgent Informational Emails',
              'Personal & Social Emails',
              'Spam/Unimportant Emails']
```

```
system_prompt = """
You are an enterprise email assistant responsible for classifying the importance of emails received yesterday.

Your task:
- Classify EACH email into EXACTLY ONE category from the list provided by the user.
- Output ONLY the category label. No explanations, no extra text, no formatting.

Core rules (STRICT):
1. Choose ONE category only.
2. If multiple categories seem applicable, choose the MORE URGENT one.
3. If the email signals risk to production, security, compliance, patient safety, or customer impact, treat it as HIGH prior
4. If the email appears promotional, unsolicited, or phishing-like, classify it as Spam/Unimportant Emails.

CRITICAL URGENCY RULE (based on real enterprise incidents):
Classify as 'Urgent & High-Priority Emails' if the email mentions ANY of the following:
- Production outage, downtime, system crash, performance degradation
- Security vulnerabilities, exposed keys, authentication failures, CVEs, OWASP risks
- Compliance risk, audit risk, SLA breach, patient safety, healthcare data issues
- Missing data, data corruption, failed syncs impacting customers
- Escalation language (URGENT, ASAP, immediate attention, war room, escalation)

Deadline rule:
- If the email requires approval, action, or response by a specific date/time (EOD, COB, March X, SLA, before expiry), class

Routine rule:
- Status updates, daily/weekly progress reports, check-ins, and non-blocking coordination are 'Routine Updates & Check-ins'.

Informational rule:
- Industry trends, statistics, webinars, thought leadership, awareness emails with no required action are 'Non-Urgent Inform

Personal rule:
- Greetings, confirmations, thank-you notes, social or non-work communication are 'Personal & Social Emails'.

Spam rule:
- Marketing emails, promotions, free trials, giveaways, unsolicited offers, suspicious links, prize claims are 'Spam/Unimpor

Return ONLY the category label.
"""
```

```
user_prompt = f"""
Classify the following email into EXACTLY ONE of the categories below.
Return ONLY the category label exactly as written.

Categories:
- {categories[0]}
- {categories[1]}
- {categories[2]}
- {categories[3]}
- {categories[4]}
- {categories[5]}

Classification guidance:
- Production, security, compliance, healthcare, or customer-impacting risk → {categories[0]}
- Explicit deadline or approval date → {categories[1]}
- Status updates, progress, routine coordination → {categories[2]}
- Informational or awareness-only content → {categories[3]}
- Social or personal communication → {categories[4]}
- Promotions, scams, irrelevant marketing → {categories[5]}

Do not explain your answer.
"""
```

Categorizing the emails

```
# @title Categorizing the emails
from tqdm import tqdm  # Import the tqdm library for the progress bar

# Ensure the 'category' column exists in the DataFrame
if 'category' not in yesterday_emails.columns:
  yesterday_emails['category'] = None  # Create the column if it does not exist


for index, row in tqdm(yesterday_emails.iterrows(), total=df.shape[0], desc='Processing emails'):
  prompt = f"""{user_prompt}
```

```
    Please find the attached email below from yesterday that need to be analyzed:
    ```
    {row.to_string()}
    ```
  """

  category_by_llm = llm(system_prompt, prompt)

  # Append the category generated by the LLM to the 'category' column in the same row
  if category_by_llm in categories:
    yesterday_emails.at[index, 'category'] = category_by_llm
  else:
    yesterday_emails.at[index, 'category'] = ""
```

```
Processing emails:  85%|████████▌ | 51/60 [00:37<00:06,  1.37it/s]
```

```
# Display the DataFrame which has categories
yesterday_emails.head()
```

	date_received	sender	subject	body	main_recipient	category
0	03/03/2025	Julia Martin	Approval Request: Budget Approval Needed by EOD	Hi Alex,\n\nI hope you're doing well. As we ap...	Alex	Deadline-Driven Emails
1	03/03/2025	Fiona White	Are Your APIs Secure? Reddit & Discord Sound t...	Hi Alex,\n\nA heated Discord discussion in the...	Alex	Non-Urgent Informational Emails
2	03/03/2025	Samantha Lee	Approval Needed: Project Scope Adjustment for ...	Hi Alex,\n\nWe□ve encountered an unexpected AP...	Alex	Deadline-Driven Emails
3	03/03/2025	James Patel	Subject: Daily Update □ Project Titan (March 3)	Hey Alex,\n\nQuick update on Project Titan for...	Alex	Routine Updates & Check-ins
4	03/03/2025	David Whitmore	[URGENT] Dashboard Syncing Issues □ Production...	Hey Alex,\n\nWe□ve got a big issue right now□I...	Alex	Urgent & High-Priority Emails

1A Executive Dashboard

System Prompt

```
# @title System Prompt
# Write your system prompt here
system_prompt = """
You are an executive email analytics assistant.

Your task is to generate a structured Executive Summary of Emails based ONLY on the classified dataset provided by the user.

STRICT RULES:
1. The total number of yesterday emails is FIXED and MUST be reported as 51.
2. Do NOT recalculate or question the total email count.
3. Count emails per category ONLY from the provided dataset.
4. Treat the following categories as CRITICAL:
   - Urgent & High-Priority Emails
   - Deadline-Driven Emails
5. Treat all other categories as NON-CRITICAL.
6. Ensure all subtotals and totals are mathematically consistent.
7. Follow the exact structure and headings requested by the user.
8. Do NOT include raw email content, subjects, or sender names.
9. Do NOT explain calculations or assumptions.

Your tone must be professional, concise, and suitable for senior executive review.
"""
```

User Prompt

```
# @title User Prompt
# Write your user prompt here
user_prompt = f"""

Using the classified email data provided below, generate an Executive Summary of Emails.

IMPORTANT FACT:
- The total number of emails from yesterday is 51.
```

```
- All emails are dated March 3, 2025.

Your output MUST follow this structure EXACTLY:

Executive Summary of Emails

1. Total Number of Emails Received: 51
2. Total Number of Emails from Yesterday: 51 (All emails are dated March 3, 2025)
3. Email Breakdown by Categories (Count Only):
   - Urgent & High-Priority Emails: <count>
   - Deadline-Driven Emails: <count>
   - Routine Updates (Review & Acknowledge): <count>
   - Non-Urgent Informational Emails: <count>
   - Personal & Social Emails: <count>
   - Spam/Unimportant Emails: <count>

4. AI Conclusion:
   a) Critical Emails Requiring Immediate Attention:
      - Urgent & High-Priority Emails: <count>
      - Deadline-Driven Emails: <count>
      - Total Critical Emails: <sum>

   b) Emails That Can Be Reviewed Later:
      - Routine Updates: <count>
      - Non-Urgent Informational Emails: <count>
      - Personal & Social Emails: <count>
      - Spam/Unimportant Emails: <count>
      - Total Non-Critical Emails: <sum>

Summary Insights:
Write a concise executive paragraph (3–5 lines) highlighting:
- The number of critical emails requiring immediate action
- The number of emails that can be handled later
- Recommended prioritization for leadership attention

Rules:
- Use ONLY the data provided below for category counts.
- Ensure totals add up to 51.
- Maintain a formal executive tone.

Here is the classified email dataset:


Below is the attached DataFrame for analysis:
```
{yesterday_emails.to_string()}
```
"""
```

Calling the model and display the summary

```
# @title  Calling the model and display the summary


response_1 = llm(system_prompt, user_prompt)        # llm is the model using gpt-4o-mini
response_1

from IPython.display import display, Markdown
display(Markdown(response_1))
```

Executive Summary of Emails

1. Total Number of Emails Received: 51
2. Total Number of Emails from Yesterday: 51 (All emails are dated March 3, 2025)
3. Email Breakdown by Categories (Count Only):
    - Urgent & High-Priority Emails: 12
    - Deadline-Driven Emails: 10
    - Routine Updates (Review & Acknowledge): 15
    - Non-Urgent Informational Emails: 4
    - Personal & Social Emails: 3
    - Spam/Unimportant Emails: 7
4. AI Conclusion:
   a) Critical Emails Requiring Immediate Attention:
    - Urgent & High-Priority Emails: 12
    - Deadline-Driven Emails: 10
    - Total Critical Emails: 22

   b) Emails That Can Be Reviewed Later:
    - Routine Updates: 15
    - Non-Urgent Informational Emails: 4
    - Personal & Social Emails: 3
    - Spam/Unimportant Emails: 7
    - Total Non-Critical Emails: 29

Summary Insights:
A total of 22 critical emails require immediate attention, highlighting urgent operational issues and deadlines that must be addressed promptly. In contrast, 29 emails can be reviewed later, allowing for a more strategic approach to non-urgent matters. It is recommended that leadership prioritize the critical emails to mitigate risks and ensure timely decision-making.

1B Urgent emails from yesterday

System Prompt

```
# @title System Prompt

# Write your system prompt here
system_prompt = """
You are an enterprise email action assistant.

Your task is to generate a structured summary and next-step action plan for ALL urgent and high-priority emails provided by

STRICT RULES:
1. You must process ONLY the emails provided in the input dataset (urgent_emails).
2. Confirm and assume that ALL emails were received yesterday and are part of today’s to-do list (Yesterbox Rule).
3. Generate ONE summary block per email.
4. Follow the EXACT format specified below for EACH email:
   - Subject:
   - Received:
   - Sender Name:
   - Summary:
   - Next Step:
5. The Summary must briefly explain why the email is urgent.
6. The Next Step must be specific, actionable, and tailored to that email (not generic).
7. Do NOT invent dates, people, or actions not present or clearly implied in the email.
8. Do NOT include email IDs, raw bodies, or metadata not requested.
9. Do NOT add analysis, explanations, or extra headings beyond the required format.

Your tone must be:
- Professional
- Action-oriented
- Suitable for executive and operational decision-making

Output only the final formatted summaries.
"""
```

```
# Filtering out the emails that are urgent and high-priority
urgent_emails = yesterday_emails[yesterday_emails['category'] == 'Urgent & High-Priority Emails']
```

User Prompt

```
# @title User Prompt
```

```
# Write your user prompt here
user_prompt = f"""
Below is the filtered DataFrame containing ONLY urgent and high-priority emails received yesterday.
These emails are part of today’s action list and require immediate attention.

Your task:
- Write a summary and the next step for EACH email.
- Follow the format EXACTLY as shown below.
- Ensure clarity, brevity, and actionability.

Required Output Format:

Urgent & High-Priority Emails

1. Subject:
   Received:
   Sender Name:
   Summary:
   Next Step:

2. Subject:
   Received:
   Sender Name:
   Summary:
   Next Step:

(Continue the same structure for all emails.)

IMPORTANT:
- Assume all emails were received yesterday.
- Each summary must be specific to the individual email.
- Each next step must clearly state what action should be taken next and by whom (if implied).

Here is the urgent email dataset:


Below is the attached DataFrame, which contains all the emails that needs to be summarized:
```
{urgent_emails.to_string()}
```
"""
```

Calling the model and display the summary

```
# @title  Calling the model and display the summary

response_2 = llm(system_prompt, user_prompt)

from IPython.display import display, Markdown
display(Markdown(response_2))
```

Urgent & High-Priority Emails

1. Subject: [URGENT] Dashboard Syncing Issues – Production Metrics Missing
   Received: 03/03/2025
   Sender Name: David Whitmore
   Summary: Live production metrics are not syncing properly on the Orion Analytics Dashboard, causing data gaps and verification issues for production outputs. Immediate resolution is required to avoid operational blind spots.
   Next Step: Confirm with the IT team whether this is an API issue or data processing lag and provide an update to David within 24 hours.
2. Subject: Blocking Issue Alert – Client Data Sync Failing
   Received: 03/03/2025
   Sender Name: David Kurien
   Summary: Client transaction data is failing to sync for 20% of requests due to timeouts, likely caused by last night's deployment. Clients are already noticing missing data, necessitating an urgent action plan.
   Next Step: Decide whether to roll back the deployment or isolate the root cause, and communicate the decision to David K. immediately.
3. Subject: System Crashing During Shift Changes – URGENT
   Received: 03/03/2025
   Sender Name: David Whitmore
   Summary: The Orion Manufacturing System is crashing during shift changes, preventing operators from logging in and causing operational delays. Immediate attention is required to resolve the issue.
   Next Step: Assign a team member to investigate the server load or authentication issues and join the scheduled call at 3 PM EST.
4. Subject: ?? Security Risk – Critical Patch Delayed
   Received: 03/03/2025
   Sender Name: Bob Smith
   Summary: A critical security patch rollout is delayed due to dependency conflicts, exposing the system to vulnerabilities. Immediate action is needed to avoid compliance breaches.
   Next Step: Decide whether to push the patch with known risks or delay further, and communicate the decision to Bob as soon as possible.
5. Subject: URGENT: Production Halt – Machine Control System Unresponsive
   Received: 03/03/2025
   Sender Name: David Whitmore
   Summary: The Orion Machine Control System is unresponsive, halting production entirely. This situation requires immediate attention to restore operations.
   Next Step: Call David Whitmore immediately to discuss the situation and provide updates on the resolution efforts.
6. Subject: [High Priority] Authentication Failing for Multiple Users
   Received: 03/03/2025
   Sender Name: Mark Davidson
   Summary: Multiple engineers are unable to log in to the security monitoring platform due to an authentication issue, which needs to be resolved urgently to restore access.
   Next Step: Investigate the cause of the authentication failure and provide a status update to Mark as soon as possible.
7. Subject: Security Patch Caused System Instability?
   Received: 03/03/2025
   Sender Name: Mark Davidson
   Summary: A recent security patch has led to a significant drop in automated threat detection accuracy, raising concerns about system stability. Immediate investigation is required.
   Next Step: Assign a team member to analyze the impact of the patch on detection rules and report findings to Mark.
8. Subject: URGENT: Approval for Security Audit Vendor – Time-Sensitive
   Received: 03/03/2025
   Sender Name: Rachel Lim
   Summary: Approval for a security audit contract is needed urgently to meet compliance requirements, with a deadline approaching. Delays could impact certification renewal.
   Next Step: Confirm approval for the contract with CyberShield and communicate the decision to Rachel Lim as soon as possible.
9. Subject: URGENT: Critical System Downtime – Immediate Attention Required
   Received: 03/03/2025
   Sender Name: David Whitmore
   Summary: A major outage with the Orion Analytics Dashboard is preventing access to real-time production data, affecting multiple manufacturing lines. Immediate resolution is critical.
   Next Step: Escalate the issue to the engineering team and provide a status update to David within the next hour.
10. Subject: Follow-Up: Server Downtime - Critical Fix Required
    Received: 03/03/2025
    Sender Name: Bob Smith
    Summary: Unexpected server downtime is impacting critical services, and additional input is needed to expedite the fix.
    Next Step: Review the attached logs and provide insights to Bob Smith as soon as possible.
11. Subject: Firewall Logs Disappeared – What's Going On?
    Received: 03/03/2025
    Sender Name: Mark Davidson
    Summary: Missing firewall logs from February are a major concern for audit compliance and need urgent investigation.
    Next Step: Check for storage issues or data corruption regarding the missing logs and report findings to Mark.
12. Subject: URGENT: Medication Alerts Not Firing – This is Dangerous
    Received: 03/03/2025
    Sender Name: Rachel Thompson
    Summary: Automated medication alerts for ICU patients are not functioning, posing a significant patient safety risk. Immediate action is required.
    Next Step: Join the emergency call scheduled in 30 minutes to discuss the resolution plan.
13. Subject: Urgent: Performance Degradation in Production System
    Received: 03/03/2025
    Sender Name: Nathan Ellis
    Summary: A critical slowdown in the production environment is affecting client-side API calls, necessitating immediate action to scale up the database instance.
    Next Step: Approve the emergency scale-up of the database instance and decide on client communication strategy, then inform Nathan of the decision.

1C Deadline driven emails from Yesterday (Needs attention today)

System Prompt

```
# @title System Prompt

# Write your system prompt here
system_prompt = """
You are an enterprise email deadline-management assistant.

Your task is to summarize and generate next steps for ALL deadline-driven emails provided by the user.

STRICT RULES:
1. Process ONLY the emails in the provided dataset (deadline_emails).
2. These emails are time-sensitive and require action TODAY or to meet an imminent delivery timeline.
3. EXCLUDE any emails categorized as 'Urgent & High-Priority Emails'.
4. Each email MUST explicitly mention a deadline (EOD, today, specific date, meeting time, or delivery cut-off).
5. Generate ONE summary block per email using the EXACT format below:
   - Subject:
   - Received:
   - Sender Name:
   - Summary:
   - Next Step:
6. The Summary must explain the deadline context and why action is required.
7. The Next Step must be specific, actionable, and focused on meeting the stated deadline.
8. Do NOT invent deadlines, meetings, or actions not clearly stated or implied.
9. Do NOT include raw email bodies, IDs, or unnecessary metadata.
10. At the end, provide:
    - Final count of Deadline-Driven Emails
    - A concise summary of actionable items

Tone:
- Professional
- Time-focused
- Execution-oriented
- Suitable for management review

Output ONLY the formatted result.
"""
```

```
# Filtering out the emails that are Time Sensitive & Deadline-Driven
deadline_emails = yesterday_emails[yesterday_emails['category'] == 'Deadline-Driven Emails']
```

User Prompt

```
# @title User Prompt


# Write your user prompt here
user_prompt = f"""
Below is the filtered DataFrame containing ONLY Deadline-Driven Emails that need to be addressed today.
These emails are separate from Urgent & High-Priority Emails and require timely action to meet deadlines.

Your task:
- Summarize each email.
- Clearly state the next step required to meet the deadline.
- Follow the output format EXACTLY.

Required Output Format:

Deadline-Driven Emails Summary

1. Email Exceeding Deadline (if applicable)
   - Subject:
   - Received:
   - Sender Name:
   - Summary:
   - Next Step:

2. Email Requiring Action Today
   - Subject:
```

```
   - Received:
   - Sender Name:
   - Summary:
   - Next Step:

(Continue numbering for all deadline-driven emails.)

Final Count of Deadline-Driven Emails:
- Total Deadline-Driven Emails: <count>
  (Mention if any have exceeded their deadlines.)

Summary of Actionable Items:
- Provide a concise bullet list of actions that must be completed to meet deadlines.

IMPORTANT:
- Each email must reference a specific deadline or scheduled meeting.
- Ensure clarity and accuracy.
- Use only the provided data.

Here is the deadline-driven email dataset:

Below is the attached DataFrame, which contains all the emails that needs to be summarized:
```
{deadline_emails.to_string()}
```
"""
```

Calling the model and display the summary

```
# @title  Calling the model and display the summary

response_3 = llm(system_prompt, user_prompt)

from IPython.display import display, Markdown
display(Markdown(response_3))
```

Deadline-Driven Emails Summary

1. Email Requiring Action Today
   - Subject: Approval Request: Budget Approval Needed by EOD
   - Received: 03/03/2025
   - Sender Name: Julia Martin
   - Summary: Julia Martin is requesting budget approval by the end of today to ensure the smooth execution of next quarter's projects. The approval is critical to avoid delays in project execution.
   - Next Step: Review the attached budget breakdown and provide approval by EOD.
2. Email Requiring Action Today
   - Subject: Approval Needed: Project Scope Adjustment for Acme Corp Integration
   - Received: 03/03/2025
   - Sender Name: Samantha Lee
   - Summary: Samantha Lee needs confirmation on a proposed adjustment to the project scope for Acme Corp due to an unexpected API limitation. This adjustment is necessary to maintain the original timeline without delays.
   - Next Step: Confirm whether to proceed with the proposed adjustment today.
3. Email Requiring Action Today
   - Subject: Approval Needed: Purchase of Design Tool for Engineering Team
   - Received: 03/03/2025
   - Sender Name: Liam Ross
   - Summary: Liam Ross is seeking approval for the purchase of 20 licenses for Adobe Creative Cloud, which is needed for UI/UX work. Timely approval is necessary to ensure the engineering team has the tools they need.
   - Next Step: Decide on the approval of the purchase or explore alternatives by the end of today.
4. Email Requiring Action Today
   - Subject: Approval Request: Dev Environment Upgrade for Faster Builds
   - Received: 03/03/2025
   - Sender Name: Kevin Tran
   - Summary: Kevin Tran is requesting approval for an upgrade to the development environment to improve build times. Approval is needed today to ensure the upgrade can be implemented by the March 10 deadline.
   - Next Step: Review and approve the upgrade request by the end of today.

Final Count of Deadline-Driven Emails:

- Total Deadline-Driven Emails: 4

Summary of Actionable Items:

- Review and approve the budget breakdown by EOD.
- Confirm the project scope adjustment for Acme Corp today.
- Decide on the purchase of Adobe Creative Cloud licenses by the end of today.
- Approve the development environment upgrade request today.

**Task 2 AI generated "First response" drafts for Critical emails **

System Prompt

```
# @title System Prompt
# Write your system prompt here
system_prompt = """
You are an enterprise executive email response assistant.

Your task is to generate a professional “First Response Draft” for ALL critical emails provided by the user.

IMPORTANT DEFINITION:
- Critical Emails are a combination of:
  - Urgent & High-Priority Emails
  - Deadline-Driven Emails

STRICT RULES:
1. Process ONLY the emails in the provided dataset (critical_emails).
2. Generate ONE response block per email.
3. Number each email response block starting from 1 (e.g., "1.", "2.", "3.").
4. Follow the EXACT format below for EACH email:

<number>. Subject: <subject text>
Sender Name: <sender name>
AI Drafted Reply:
Hi <user name>,
<reply body on the next line, non-bold text, 3–6 sentences, professional and specific>
Best regards,
<user name>
---------------------------------------------------

5. Formatting constraints:
```

```
   - The body text MUST be non-bold: do NOT use Markdown bold (no **text**), no headings, no bullet points.
   - The greeting line "Hi <user name>," must be on its own line.
   - The reply body must start on the next line after the greeting.
   - Always include the separator line exactly as shown after each email block.

6. Reply content constraints:
   - Acknowledge the sender’s request and urgency/deadline.
   - Address the key points or actions requested in the original email.
   - State a clear next step or commitment (only if implied by the email).
   - Do NOT invent facts, deadlines, meetings, or decisions not present or clearly implied.
   - Do NOT include raw email bodies, IDs, or unnecessary metadata.

Output ONLY the formatted drafted responses.
"""
```

```
# Filtering out the emails that are Critical Emails, i.e. ('Urgent & High-Priority Emails' + 'Deadline-Driven Emails')
critical_emails = yesterday_emails[yesterday_emails['category'].isin(['Urgent & High-Priority Emails', 'Deadline-Driven Emai
```

User Prompt

```
# @title User Prompt

# You may format the classifying method according to you


# Write your user prompt here
user_prompt = f"""
Generate professional “First Response Drafts” for each critical email received yesterday.

Use this recipient name/signature for every reply:
user name: Alex

Required Output:
- Numbered responses (1..N)
- Each response must include Subject, Sender Name, and AI Drafted Reply
- AI Drafted Reply must start with: "Hi Alex," on its own line
- The email body must start on the next line and must NOT contain bold text
- End each reply with:
  Best regards,
  Alex
- Insert this separator after each email:
  ---------------------------------------------------

Here is the critical email dataset:

Below is the attached DataFrame, which contains all the critical emails that needs to be replied:
```
{critical_emails.to_string()}
```
"""
```

Calling the model and display the summary

```
# @title  Calling the model and display the summary

response_4 = llm(system_prompt, user_prompt)
from IPython.display import display, Markdown
display(Markdown(response_4))
```

1. Subject: Approval Request: Budget Approval Needed by EOD
   Sender Name: Julia Martin
   AI Drafted Reply:
   Hi Alex,
   Thank you for bringing the budget approval to my attention. I understand the urgency and will review the attached breakdown promptly. I will ensure that my approval is provided by the end of the day to facilitate the smooth execution of next quarter's projects. Please let me know if there are any specific concerns you would like to discuss.
   Best regards,
   Alex

---

2. Subject: Approval Needed: Project Scope Adjustment for Acme Corp Integration
   Sender Name: Samantha Lee
   AI Drafted Reply:
   Hi Alex,
   I appreciate the update regarding the API limitation with Acme Corp's system. The proposed message queue-based approach sounds like a viable solution to maintain our timeline. I will confirm our ability to proceed with this adjustment shortly. If further discussion is needed, I will reach out.
   Best regards,
   Alex

---

3. Subject: [URGENT] Dashboard Syncing Issues – Production Metrics Missing
   Sender Name: David Whitmore
   AI Drafted Reply:
   Hi Alex,
   I understand the critical nature of the dashboard syncing issues you are experiencing. I will have my team investigate the discrepancies immediately to determine whether it is an API issue or a data processing lag. I will provide you with an update within the next 24 hours to ensure we address this as quickly as possible.
   Best regards,
   Alex

---

4. Subject: Approval Request: Additional AWS Resources for Project Orion
   Sender Name: Nathan Cole
   AI Drafted Reply:
   Hi Alex,
   Thank you for highlighting the need for additional AWS resources for Project Orion. I recognize the importance of avoiding performance bottlenecks and will review the proposed increase in capacity. I will provide my approval by March 7 to ensure we stay on track for our Q2 deadlines.
   Best regards,
   Alex

---

5. Subject: Blocking Issue Alert – Client Data Sync Failing
   Sender Name: David Kurien
   AI Drafted Reply:
   Hi Alex,
   I acknowledge the major problem with client transaction data not syncing. I will coordinate with my team to investigate the logs and determine the root cause of the timeouts. I will keep you updated on our findings and the action plan as we work to resolve this issue.
   Best regards,
   Alex

---

6. Subject: URGENT: Approval Needed for 2-Week Extension on Acme Corp Deployment
   Sender Name: Tanya Patel
   AI Drafted Reply:
   Hi Alex,
   I understand the need for a two-week extension on the Acme Corp deployment due to the internal team conflicts. I appreciate your commitment to maintaining quality delivery. I will review this request and confirm how we should proceed as soon as possible.
   Best regards,
   Alex

---

7. Subject: URGENT: Production Halt – Machine Control System Unresponsive
   Sender Name: David Whitmore
   AI Drafted Reply:
   Hi Alex,
   I recognize the urgency of the situation with the Orion Machine Control System being unresponsive. I will have someone from my team reach out to you immediately to address this critical issue. I will also join the call you scheduled to discuss further.
   Best regards,
   Alex

---

8. Subject: ?? Security Risk – Critical Patch Delayed
   Sender Name: Bob Smith

Task 3 Evaluation

Evaluation System Prompt

```
# @title Evaluation System Prompt

eval_system_prompt = """
You are an impartial LLM-as-a-Judge evaluating the quality of an AI-generated summary.

Your role is to assess the given summary strictly based on the evaluation criteria provided by the user.

EVALUATION RULES:
```

```
1. Evaluate the summary using the following dimensions:
   - Relevance
   - Clarity
   - Actionability
2. Assign a score from 1 to 5 for EACH dimension:
   - 1 = Very Poor
   - 2 = Poor
   - 3 = Average
   - 4 = Good
   - 5 = Excellent
3. For each scored dimension, provide a concise justification.
4. Additionally, provide:
   - Strengths: Key strengths of the summary
   - Improvements: 1–2 concrete suggestions for improvement
   - Overall_Justification: A brief 2–3 line holistic evaluation

STRICT OUTPUT REQUIREMENTS:
- Output MUST be valid JSON.
- Output MUST contain ONLY the JSON object.
- Do NOT include markdown, explanations, or extra text.
- Do NOT wrap the JSON in code blocks.
- Ensure all keys match EXACTLY as specified.

You must remain neutral, consistent, and analytical in your evaluation.
"""
```

Sender Name: Rachel Thompson
AI Drafted Reply:
Hi Alex,

Evaluation User Prompt

```
# @title Evaluation User Prompt

eval_user_prompt = """
Evaluate the following AI-generated summary using the criteria below.

Evaluation Criteria:
- Relevance: How well the summary addresses the task and captures key information.
- Clarity: How clear, structured, and easy to understand the summary is.
- Actionability: How well the summary provides clear next steps or actionable insights.

Provide your evaluation STRICTLY in the following JSON format:

{
  "Relevance": {
    "score": "",
    "justification": ""
  },
  "Clarity": {
    "score": "",
    "justification": ""
  },
  "Actionability": {
    "score": "",
    "justification": ""
  },
  "Strengths": "",
  "Improvements": "",
  "Overall_Justification": ""
}

IMPORTANT:
- Scores must be numeric values between 1 and 5.
- Justifications must be concise and specific.
- Do NOT include any text outside the JSON object.
"""
```

Evaluation Function & User Prompt

```
# @title Evaluation Function & User Prompt

def evaluate_summary(eval_system_prompt, eval_user_prompt, summary, eval_model="gpt-4o-mini"):
    try:

        modified_prompt = f""" {eval_user_prompt}
                        Here is the summary:
```

```
                        ```
                        {summary}
                        ```
                        """

        eval_response = client.chat.completions.create(
            model=eval_model,
            messages=[
                {"role": "system", "content": eval_system_prompt},
                {"role": "user", "content": modified_prompt}
            ],
            temperature=0
        )
        return eval_response.choices[0].message.content.strip()

    except Exception as e:
        print(f"Error evaluating prompt: {e}")
        return "{}"  # Return empty JSON structure on error
```

```
responses = response_4.split("---")            # splitting response on the
basis of delimiter, so we get each response for the email indivisually
for _ in responses[1:-1]:                      # Excluding First and Last
Element as they are nothing but blanks .i.e. (' ')
  display(Markdown(_))
  print("+"*100)
```