1. Message to Teams

import requests

import json

import os

from datetime import datetime

def handler(pd: "pipedream"):

"""

Send comprehensive Teams notification with full business intelligence

"""

print("📢 Preparing comprehensive Teams notification...")

# Get CRM creation data

if "Lead\_in\_Zoho\_CRM" not in pd.steps:

return {"error": "CRM step not found"}

crm\_step = pd.steps["Lead\_in\_Zoho\_CRM"]

# Access CRM data

crm\_data = None

if isinstance(crm\_step, dict):

if "event" in crm\_step:

crm\_data = crm\_step["event"]

elif "$return\_value" in crm\_step:

crm\_data = crm\_step["$return\_value"]

elif "success" in crm\_step:

crm\_data = crm\_step

if not crm\_data or not crm\_data.get("success", False):

skip\_reason = crm\_data.get("error", "CRM creation failed") if crm\_data else "No CRM data"

return {"skipped": True, "reason": skip\_reason}

# Extract comprehensive data

lead\_id = crm\_data.get("lead\_id", "Unknown")

comprehensive\_data = crm\_data.get("comprehensive\_data", {})

contact\_info = comprehensive\_data.get("contact\_info", {})

business\_intel = comprehensive\_data.get("business\_intelligence", {})

project\_intel = comprehensive\_data.get("project\_intelligence", {})

sales\_intel = comprehensive\_data.get("sales\_intelligence", {})

print(f"📧 Creating comprehensive notification for: {contact\_info.get('email', 'Unknown')}")

print(f"🆔 Lead ID: {lead\_id}")

# Teams webhook URL

teams\_webhook\_url = os.environ.get('TEAMS\_WEBHOOK\_URL')

if not teams\_webhook\_url:

return {"error": "Teams webhook URL not configured"}

# Determine notification priority and styling

priority\_info = determine\_notification\_priority(project\_intel, sales\_intel, business\_intel)

# Create comprehensive Teams message

teams\_message = create\_comprehensive\_teams\_message(

contact\_info, business\_intel, project\_intel, sales\_intel,

lead\_id, priority\_info

)

try:

print("🚀 Sending comprehensive Teams notification...")

response = requests.post(

teams\_webhook\_url,

headers={"Content-Type": "application/json"},

json=teams\_message,

timeout=30

)

print(f"📊 Teams API Response: {response.status\_code}")

if response.status\_code == 200:

print("✅ Comprehensive Teams notification sent successfully!")

notification\_summary = create\_notification\_summary(

contact\_info, business\_intel, project\_intel, sales\_intel,

lead\_id, priority\_info

)

return {

"success": True,

"message": "Comprehensive Teams notification sent",

"notification\_summary": notification\_summary,

"priority\_level": priority\_info["priority\_text"],

"business\_intelligence\_included": True

}

else:

error\_text = response.text[:200] if response.text else "No error details"

return {

"success": False,

"error": f"Teams API Error: {response.status\_code}",

"error\_details": error\_text

}

except Exception as e:

return {"success": False, "error": f"Unexpected error: {str(e)}"}

def determine\_notification\_priority(project\_intel, sales\_intel, business\_intel):

"""Determine notification priority based on comprehensive intelligence"""

urgency = project\_intel.get("urgency\_level", "none")

project\_size = project\_intel.get("estimated\_project\_size", "unknown")

buying\_stage = sales\_intel.get("buying\_stage", "unknown")

close\_probability = sales\_intel.get("close\_probability", "unknown")

deal\_size = sales\_intel.get("estimated\_deal\_size", "")

priority\_score = 0

# Urgency scoring

urgency\_scores = {"critical": 40, "high": 30, "medium": 20, "low": 10, "none": 0}

priority\_score += urgency\_scores.get(urgency, 0)

# Project size scoring

size\_scores = {"enterprise": 25, "large": 20, "medium": 15, "small": 10, "unknown": 5}

priority\_score += size\_scores.get(project\_size, 0)

# Buying stage scoring

stage\_scores = {"ready-to-buy": 25, "decision": 20, "evaluation": 15, "research": 10, "unknown": 0}

priority\_score += stage\_scores.get(buying\_stage, 0)

# Close probability scoring

prob\_scores = {"high": 15, "medium": 10, "low": 5, "unknown": 0}

priority\_score += prob\_scores.get(close\_probability, 0)

# Deal size bonus (if contains high numbers)

if any(indicator in deal\_size.lower() for indicator in ["100k", "million", "large", "enterprise"]):

priority\_score += 15

# Determine priority level

if priority\_score >= 80:

return {

"priority\_text": "🔥 CRITICAL PRIORITY",

"urgency\_emoji": "🔥",

"theme\_color": "DC143C", # Crimson red

"priority\_level": "critical"

}

elif priority\_score >= 60:

return {

"priority\_text": "⚡ HIGH PRIORITY",

"urgency\_emoji": "⚡",

"theme\_color": "FF4500", # Orange red

"priority\_level": "high"

}

elif priority\_score >= 40:

return {

"priority\_text": "⭐ MEDIUM PRIORITY",

"urgency\_emoji": "⭐",

"theme\_color": "FF8C00", # Dark orange

"priority\_level": "medium"

}

else:

return {

"priority\_text": "💡 STANDARD PRIORITY",

"urgency\_emoji": "💡",

"theme\_color": "4169E1", # Royal blue

"priority\_level": "standard"

}

def create\_comprehensive\_teams\_message(contact\_info, business\_intel, project\_intel, sales\_intel, lead\_id, priority\_info):

"""Create comprehensive Teams message card"""

company\_name = contact\_info.get("company", "Unknown Company")

contact\_name = contact\_info.get("full\_name", "Unknown Contact")

return {

"@type": "MessageCard",

"@context": "http://schema.org/extensions",

"themeColor": priority\_info["theme\_color"],

"summary": f"🎯 New Qualified Lead: {contact\_name} from {company\_name}",

"sections": [

{

"activityTitle": f"{priority\_info['urgency\_emoji']} AI-QUALIFIED LEAD DETECTED",

"activitySubtitle": f"{priority\_info['priority\_text']} | Business Intelligence Available",

"activityImage": "https://cdn-icons-png.flaticon.com/512/3135/3135715.png",

"facts": create\_contact\_facts(contact\_info, lead\_id),

"markdown": True

},

{

"activityTitle": "🏢 BUSINESS INTELLIGENCE",

"facts": create\_business\_intelligence\_facts(business\_intel),

"markdown": True

},

{

"activityTitle": "📋 PROJECT INTELLIGENCE",

"facts": create\_project\_intelligence\_facts(project\_intel),

"markdown": True

},

{

"activityTitle": "💼 SALES INTELLIGENCE",

"facts": create\_sales\_intelligence\_facts(sales\_intel),

"markdown": True

},

{

"activityTitle": "🎯 RECOMMENDED ACTIONS",

"text": create\_recommended\_actions(project\_intel, sales\_intel, business\_intel)

}

],

"potentialAction": create\_action\_buttons(contact\_info, lead\_id, priority\_info)

}

def create\_contact\_facts(contact\_info, lead\_id):

"""Create contact information facts"""

return [

{"name": "👤 Contact Name", "value": contact\_info.get("full\_name", "Unknown")},

{"name": "📧 Email Address", "value": contact\_info.get("email", "Unknown")},

{"name": "📞 Phone", "value": contact\_info.get("primary\_phone", "Not provided")},

{"name": "📱 Mobile", "value": contact\_info.get("mobile\_phone", "Not provided")},

{"name": "🏢 Company", "value": contact\_info.get("company", "Unknown")},

{"name": "🏭 Industry", "value": contact\_info.get("industry", "Unknown")},

{"name": "💼 Title", "value": contact\_info.get("title", "Not specified")},

{"name": "🌐 Website", "value": contact\_info.get("website", "Not provided")},

{"name": "🆔 CRM Lead ID", "value": str(lead\_id)},

{"name": "📅 Created", "value": datetime.now().strftime("%Y-%m-%d %H:%M:%S")}

]

def create\_business\_intelligence\_facts(business\_intel):

"""Create business intelligence facts"""

return [

{"name": "🏢 Company Size", "value": business\_intel.get("company\_size", "Unknown")},

{"name": "👨‍💼 Decision Maker", "value": business\_intel.get("decision\_maker\_title", "Unknown")},

{"name": "📊 Project Type", "value": business\_intel.get("project\_type", "Not specified")},

{"name": "📈 Record Volume", "value": business\_intel.get("record\_volume", "Not specified")},

{"name": "🔧 Services Needed", "value": format\_list\_field(business\_intel.get("services\_needed", []))},

{"name": "⚖️ Compliance", "value": format\_list\_field(business\_intel.get("compliance\_requirements", []))},

{"name": "💻 Technology", "value": format\_list\_field(business\_intel.get("technology\_requirements", []))},

{"name": "💰 Budget Indicators", "value": business\_intel.get("budget\_indicators", "Not mentioned")},

{"name": "⏰ Timeline", "value": business\_intel.get("timeline\_urgency", "Not specified")},

{"name": "🏆 Success Metrics", "value": format\_list\_field(business\_intel.get("success\_metrics", []))}

]

def create\_project\_intelligence\_facts(project\_intel):

"""Create project intelligence facts"""

return [

{"name": "📏 Project Size", "value": project\_intel.get("estimated\_project\_size", "Unknown").title()},

{"name": "🚨 Urgency Level", "value": project\_intel.get("urgency\_level", "Unknown").upper()},

{"name": "🧩 Complexity", "value": project\_intel.get("complexity\_level", "Unknown").title()},

{"name": "📋 Compliance Level", "value": project\_intel.get("compliance\_level", "Unknown").title()},

{"name": "🤖 Automation Potential", "value": project\_intel.get("automation\_potential", "Unknown").title()}

]

def create\_sales\_intelligence\_facts(sales\_intel):

"""Create sales intelligence facts"""

return [

{"name": "🛒 Buying Stage", "value": sales\_intel.get("buying\_stage", "Unknown").title()},

{"name": "💳 Budget Authority", "value": sales\_intel.get("budget\_authority", "Unknown").title()},

{"name": "🎯 Decision Influence", "value": sales\_intel.get("decision\_influence", "Unknown").title()},

{"name": "🏁 Competitive Situation", "value": sales\_intel.get("competitive\_situation", "Unknown").title()},

{"name": "📊 Close Probability", "value": sales\_intel.get("close\_probability", "Unknown").upper()},

{"name": "💵 Estimated Deal Size", "value": sales\_intel.get("estimated\_deal\_size", "Not estimated")}

]

def create\_recommended\_actions(project\_intel, sales\_intel, business\_intel):

"""Create recommended actions based on intelligence"""

actions = []

urgency = project\_intel.get("urgency\_level", "none")

buying\_stage = sales\_intel.get("buying\_stage", "unknown")

close\_probability = sales\_intel.get("close\_probability", "unknown")

if urgency in ["critical", "high"]:

actions.append("🚨 \*\*IMMEDIATE ACTION REQUIRED\*\* - High urgency project")

actions.append("📞 Call within 2 hours")

elif urgency == "medium":

actions.append("⏰ Contact within 24 hours")

else:

actions.append("📧 Send follow-up email within 2 business days")

if buying\_stage == "ready-to-buy":

actions.append("💰 Prepare proposal immediately - ready to buy")

elif buying\_stage == "decision":

actions.append("📋 Send detailed proposal and pricing")

elif buying\_stage == "evaluation":

actions.append("📊 Provide case studies and references")

if close\_probability == "high":

actions.append("🎯 Assign senior sales rep - high close probability")

if business\_intel.get("competitive\_situation") == "rfp":

actions.append("📄 Request RFP documents and requirements")

compliance\_reqs = business\_intel.get("compliance\_requirements", [])

if compliance\_reqs:

actions.append(f"⚖️ Prepare compliance documentation: {', '.join(compliance\_reqs)}")

if not actions:

actions.append("📞 Standard follow-up within 2 business days")

return "\\n".join(f"• {action}" for action in actions)

def create\_action\_buttons(contact\_info, lead\_id, priority\_info):

"""Create action buttons for Teams message"""

email = contact\_info.get("email", "")

phone = contact\_info.get("primary\_phone", "")

company = contact\_info.get("company", "")

# Create mailto link with professional template

subject = f"Re: Your Data Entry Services Inquiry - {company}"

body = f"Dear {contact\_info.get('full\_name', 'Valued Customer')},\\n\\nThank you for your inquiry about our data entry services. I'm reaching out to discuss your project requirements and how we can help.\\n\\nI'd like to schedule a brief call to better understand your needs and provide you with a tailored solution.\\n\\nBest regards,\\nPerfect Data Entry Team"

mailto\_url = f"mailto:{email}?subject={subject}&body={body}".replace(" ", "%20").replace("\n", "%0D%0A")

actions = [

{

"@type": "OpenUri",

"name": "🔗 Open in Zoho CRM",

"targets": [{"os": "default", "uri": f"https://crm.zoho.com/crm/ShowEntityInfo.do?module=Leads&id={lead\_id}"}]

},

{

"@type": "OpenUri",

"name": "📧 Send Professional Reply",

"targets": [{"os": "default", "uri": mailto\_url}]

}

]

if phone and phone != "Not provided":

# Add call button if phone available

actions.append({

"@type": "OpenUri",

"name": f"📞 Call {phone}",

"targets": [{"os": "default", "uri": f"tel:{phone.replace(' ', '').replace('(', '').replace(')', '').replace('-', '')}"}]

})

# Add priority-based actions

if priority\_info["priority\_level"] in ["critical", "high"]:

actions.append({

"@type": "OpenUri",

"name": "🚨 Escalate to Senior Sales",

"targets": [{"os": "default", "uri": f"mailto:sales-manager@perfectdataentry.com?subject=URGENT: High Priority Lead - {company}&body=High priority lead requires immediate attention. Lead ID: {lead\_id}"}]

})

return actions

def format\_list\_field(field\_list):

"""Format list fields for display"""

if not field\_list or not isinstance(field\_list, list):

return "Not specified"

if len(field\_list) == 1:

return field\_list[0]

elif len(field\_list) <= 3:

return ", ".join(field\_list)

else:

return f"{', '.join(field\_list[:3])} and {len(field\_list) - 3} more"

def create\_notification\_summary(contact\_info, business\_intel, project\_intel, sales\_intel, lead\_id, priority\_info):

"""Create notification summary for response"""

return {

"timestamp": datetime.now().isoformat(),

"lead\_id": lead\_id,

"contact\_name": contact\_info.get("full\_name", "Unknown"),

"contact\_email": contact\_info.get("email", "Unknown"),

"company": contact\_info.get("company", "Unknown"),

"industry": contact\_info.get("industry", "Unknown"),

"priority\_level": priority\_info["priority\_level"],

"urgency\_level": project\_intel.get("urgency\_level", "unknown"),

"project\_size": project\_intel.get("estimated\_project\_size", "unknown"),

"buying\_stage": sales\_intel.get("buying\_stage", "unknown"),

"close\_probability": sales\_intel.get("close\_probability", "unknown"),

"business\_intelligence\_available": True,

"recommended\_action": "immediate\_contact" if priority\_info["priority\_level"] in ["critical", "high"] else "standard\_follow\_up"

}