import pandas as pd

from openpyxl import load\_workbook

from datetime import datetime, timedelta

import logging

import os

import glob

from tvr\_processor import extract\_tvr\_data # ✅ IMPORT the TVR extractor

logger = logging.getLogger(\_\_name\_\_)

def safe\_get\_cell(df, row, col, default=0):

try:

value = df.iloc[row, col]

return value if not pd.isna(value) else default

except IndexError:

logger.warning(f"Warning: Index [{row},{col}] out of bounds for dataframe with shape {df.shape}")

return default

def safe\_set\_cell(ws, cell\_ref, value):

for merged\_range in ws.merged\_cells.ranges:

if cell\_ref in merged\_range:

cell\_ref = merged\_range.start\_cell.coordinate

break

ws[cell\_ref] = value

return cell\_ref

def process\_excel\_data(input\_a\_path, input\_b\_path, skeleton\_path, output\_path):

logger.info(f"Process started on {datetime.now().strftime('%A, %B %d, %Y at %H:%M:%S')}")

# File checks (skip TVR Output if missing)

for path, name in [(input\_a\_path, "Non Cricket Input"), (skeleton\_path, "Skeleton")]:

if not os.path.exists(path):

error\_msg = f"{name} file not found at path: {path}"

logger.error(error\_msg)

raise FileNotFoundError(error\_msg)

tvr\_file\_exists = os.path.exists(input\_b\_path)

if not tvr\_file\_exists:

logger.warning(f"TVR Output file not found at path: {input\_b\_path}. Continuing without TVR data.")

try:

# Load data

property\_details = pd.read\_excel(input\_a\_path, sheet\_name="Property Details", header=None)

channel\_platform = pd.read\_excel(input\_a\_path, sheet\_name="Channel & Platform Details", header=None)

program\_performance = pd.read\_excel(input\_a\_path, sheet\_name="Program Performance", header=None)

input\_b = pd.read\_excel(input\_b\_path, header=None) if tvr\_file\_exists else None

except Exception as e:

logger.error(f"Error loading input files: {str(e)}")

raise

wb = load\_workbook(skeleton\_path)

sheet1 = wb[wb.sheetnames[0]]

sheet2 = wb[wb.sheetnames[1]]

# Extraction logic (unchanged)

prop\_b1 = safe\_get\_cell(property\_details, 0, 1, "")

prop\_b8 = safe\_get\_cell(property\_details, 7, 1, "")

prop\_b14 = safe\_get\_cell(property\_details, 13, 1, "")

prop\_b29 = safe\_get\_cell(property\_details, 28, 1, "")

prop\_a29 = safe\_get\_cell(property\_details, 28, 0, "")

prop\_b3 = safe\_get\_cell(property\_details, 2, 1, "")

prop\_b4 = safe\_get\_cell(property\_details, 3, 1, "")

prop\_b7 = safe\_get\_cell(property\_details, 6, 1, "")

prop\_b9 = safe\_get\_cell(property\_details, 8, 1, "")

prop\_b10 = safe\_get\_cell(property\_details, 9, 1, "")

prop\_b11 = safe\_get\_cell(property\_details, 10, 1, "")

prop\_b12 = safe\_get\_cell(property\_details, 11, 1, 2)

prop\_b13 = safe\_get\_cell(property\_details, 12, 1, 1)

prop\_b20 = safe\_get\_cell(property\_details, 19, 1, "")

prop\_b21 = safe\_get\_cell(property\_details, 20, 1, "")

prop\_b22 = safe\_get\_cell(property\_details, 21, 1, "")

prop\_b23 = safe\_get\_cell(property\_details, 22, 1, "")

prop\_b26 = safe\_get\_cell(property\_details, 25, 1, "")

prop\_b27 = safe\_get\_cell(property\_details, 26, 1, "")

prop\_b28 = safe\_get\_cell(property\_details, 27, 1, "")

prop\_b32 = safe\_get\_cell(property\_details, 31, 1, 0)

channel\_b5 = safe\_get\_cell(channel\_platform, 4, 1, "")

channel\_c5 = safe\_get\_cell(channel\_platform, 4, 2, "")

channel\_c6 = safe\_get\_cell(channel\_platform, 5, 2, "")

channel\_c7 = safe\_get\_cell(channel\_platform, 6, 2, "")

channel\_c8 = safe\_get\_cell(channel\_platform, 7, 2, "")

channel\_c9 = safe\_get\_cell(channel\_platform, 8, 2, "")

channel\_e9 = safe\_get\_cell(channel\_platform, 8, 4, 0)

channel\_e10 = safe\_get\_cell(channel\_platform, 9, 4, 0)

channel\_g5 = safe\_get\_cell(channel\_platform, 4, 6, 0)

channel\_g6 = safe\_get\_cell(channel\_platform, 5, 6, 0)

channel\_g7 = safe\_get\_cell(channel\_platform, 6, 6, 0)

channel\_g8 = safe\_get\_cell(channel\_platform, 7, 6, 0)

channel\_o5 = safe\_get\_cell(channel\_platform, 4, 14, 0)

channel\_o6 = safe\_get\_cell(channel\_platform, 5, 14, 0)

channel\_o7 = safe\_get\_cell(channel\_platform, 6, 14, 0)

channel\_o8 = safe\_get\_cell(channel\_platform, 7, 14, 0)

channel\_j9 = safe\_get\_cell(channel\_platform, 8, 9, 0)

channel\_j10 = safe\_get\_cell(channel\_platform, 9, 9, 0)

channel\_k9 = safe\_get\_cell(channel\_platform, 8, 10, 0)

channel\_k10 = safe\_get\_cell(channel\_platform, 9, 10, 0)

channel\_l9 = safe\_get\_cell(channel\_platform, 8, 11, 0)

channel\_l10 = safe\_get\_cell(channel\_platform, 9, 11, 0)

program\_l11 = safe\_get\_cell(program\_performance, 10, 11, 0)

program\_l12 = safe\_get\_cell(program\_performance, 11, 11, 0)

program\_f11 = safe\_get\_cell(program\_performance, 10, 5, 0)

program\_g11 = safe\_get\_cell(program\_performance, 10, 6, 0)

program\_f12 = safe\_get\_cell(program\_performance, 11, 5, 0)

program\_g12 = safe\_get\_cell(program\_performance, 11, 6, 0)

current\_year = datetime.now().year

safe\_set\_cell(sheet2, 'B2', f"{prop\_b1} - {current\_year} Driven By: {prop\_b29}")

safe\_set\_cell(sheet2, 'C5', prop\_b3)

if isinstance(prop\_b8, str):

try:

prop\_b8 = datetime.strptime(prop\_b8, "%d %B %Y")

except Exception:

logger.warning(f"Could not parse date from prop\_b8: {prop\_b8}, using today's date.")

prop\_b8 = datetime.now()

campaign\_end\_date = prop\_b8 + timedelta(weeks=int(prop\_b14) if prop\_b14 else 0)

start\_month = prop\_b8.strftime("%b'%y")

end\_month = campaign\_end\_date.strftime("%b'%y")

result = f"{start\_month} - {end\_month}"

# ...rest of your logic for ER/CPRP, writing to sheets, etc. (unchanged)...

# D41 and F41 formulas

safe\_set\_cell(sheet1, 'D41', "='DBD One Pager-with Eval.'!D45")

safe\_set\_cell(sheet1, 'F41', "='DBD One Pager-with Eval.'!E45")

# ✅✅✅ Call TVR Extractor and write to H30 and I30:

tvrs = None

if tvr\_file\_exists:

try:

tvrs = extract\_tvr\_data(input\_a\_path)

except Exception as e:

logger.warning(f"TVR extraction failed: {e}")

tvrs = None

else:

logger.info("Skipping TVR extraction as TVR file is missing.")

if tvrs and len(tvrs) >= 2:

safe\_set\_cell(sheet2, 'I30', tvrs[0])

safe\_set\_cell(sheet2, 'H30', tvrs[1])

logger.info(f"✅ TVRs written: H30={tvrs[0]}, I30={tvrs[1]}")

else:

logger.warning("⚠️ No TVRs returned to write in H30, I30.")

# ✅ Save workbook

os.makedirs(os.path.dirname(os.path.abspath(output\_path)), exist\_ok=True)

wb.save(output\_path)

logger.info(f"✅ Process finished. Output saved to {output\_path}")

print(f"✅ Detailed Package file generated successfully")

if \_\_name\_\_ == "\_\_main\_\_":

non\_cricket\_folder = "input/non\_cricket\_input/"

input\_b = "input/TVR Output.xlsx"

skeleton = "input/Skeleton Output.xlsx"

output = "output/Completed\_Output.xlsx"

non\_cricket\_files = glob.glob(os.path.join(non\_cricket\_folder, "\*.xlsx"))

if not non\_cricket\_files:

raise FileNotFoundError(f"No Non Cricket Input file found in {non\_cricket\_folder}")

input\_a = non\_cricket\_files[0] # Pick first match

print(f"Using Non Cricket Input