```
import pandas_1as pd
from datetime import datetime
import tkinter as tk
from tkinter import messagebox
import os
import numpy as np

def identify_and_remove_empty_rows(df, threshold=0.7):
    """
    識別並刪除包含大量空值(nan)的行
    threshold: 空值比例閾值，超過此比例的行被視為空行
    返回: (清理後的df, 刪除的行索引列表)
    """
    empty_rows = []

    print("=== 分析每行的空值情況 ===")
    for i in range(len(df)):
        row = df.iloc[i]
        # 計算nan值的比例
        nan_count = row.isna().sum()
        total_count = len(row)
        nan_ratio = nan_count / total_count

        # 顯示前10列的內容
        preview = row.iloc[:10].tolist()
        print(f"第{i+1}行: nan比例={nan_ratio:.2f} ({nan_count}/{total_count}), 前10列:
{preview}")

        if nan_ratio >= threshold:
            empty_rows.append(i)
            print(f"  -> 標記為空行（將刪除）")

    if empty_rows:
        print(f"\n發現 {len(empty_rows)} 個空行，索引: {empty_rows}")
        df_cleaned = df.drop(df.index[empty_rows]).reset_index(drop=True)
        print(f"刪除空行後形狀: {df_cleaned.shape}")
        return df_cleaned, empty_rows
    else:
        print("沒有發現需要刪除的空行")
        return df, []

def process_reports():
    """
    處理兩個報告文件並生成最終結果
    從指定路徑讀取文件
    """

    # 設定文件路徑
```

```
today = datetime.now().strftime('%Y%m%d')
report1_path = f"C:\\Users\\vic25\\R1_report_{today}.xlsx"
report2_path = "C:\\Users\\vic25\\report_2.xlsx"

print(f"查找文件:")
print(f"Report 1: {report1_path}")
print(f"Report 2: {report2_path}")

# 檢查文件是否存在
if not os.path.exists(report1_path):
    error_msg = f"找不到文件: {report1_path}\n請確認SAP已生成今天的報告文件。"
    print(error_msg)
    root = tk.Tk()
    root.withdraw()
    messagebox.showerror("文件錯誤", error_msg)
    root.destroy()
    return None

if not os.path.exists(report2_path):
    error_msg = f"找不到文件: {report2_path}\n請確認report_2.xlsx文件存在。"
    print(error_msg)
    root = tk.Tk()
    root.withdraw()
    messagebox.showerror("文件錯誤", error_msg)
    root.destroy()
    return None

# 讀取報告1和報告2
print("讀取報告文件...")
try:
    # 先以無標題方式讀取，檢查結構
    df1_raw = pd.read_excel(report1_path, header=None)
    df2 = pd.read_excel(report2_path)
except Exception as e:
    error_msg = f"讀取Excel文件時出錯: {str(e)}"
    print(error_msg)
    root = tk.Tk()
    root.withdraw()
    messagebox.showerror("讀取錯誤", error_msg)
    root.destroy()
    return None

print(f"Report 1 原始形狀: {df1_raw.shape}")
print(f"Report 2 形狀: {df2.shape}")

# 識別並刪除空行
df1_cleaned, removed_rows = identify_and_remove_empty_rows(df1_raw, threshold=0.7)
```

```
# 重新讀取，但這次正確設置標題行
print("\n=== 重新設置標題行 ===")
if removed_rows:
    # 如果刪除了行，需要調整header位置
    # 假設原本第一行是標題，如果第一行被刪除了，那麼現在第一行應該是標題
    header_row = 0
else:
    # 如果沒有刪除行，檢查第一行是否是標題
    header_row = 0

# 使用清理後的數據，手動設置標題
df1 = df1_cleaned.copy()

# 設置列名為第一行，然後刪除第一行
if len(df1) > 0:
    # 檢查第一行是否看起來像標題行
    first_row = df1.iloc[0]
    string_count = sum(1 for val in first_row if pd.notna(val) and isinstance(val, str))

    if string_count > len(first_row) * 0.3:  # 如果超過30%是字符串，當作標題行
        print("第一行識別為標題行")
        column_names = []
        for i, val in enumerate(first_row):
            if pd.notna(val) and str(val).strip() != '':
                column_names.append(str(val).strip())
            else:
                column_names.append(f'Column_{i+1}')

        df1.columns = column_names
        df1 = df1.iloc[1:].reset_index(drop=True)  # 刪除標題行
        print(f"設置標題後的形狀: {df1.shape}")
        print(f"標題行: {column_names[:10]}")
    else:
        print("第一行不像標題行，使用通用列名")
        df1.columns = [f'Column_{i+1}' for i in range(df1.shape[1])]

print("\n=== 清理後的數據預覽 ===")
for i in range(min(3, len(df1))):
    print(f"第{i+1}行前10列: {df1.iloc[i, :10].tolist()}")

# 創建查找字典 - 從report2建立Installation到Order Status的映射
print("\n建立查找映射...")
if df2.shape[1] < 10:
    error_msg = f"Report2列數不足，期望至少10列，實際{df2.shape[1]}列"
    print(error_msg)
    root = tk.Tk()
    root.withdraw()
    messagebox.showerror("數據錯誤", error_msg)
```

```
    root.destroy()
    return None

lookup_dict = dict(zip(df2.iloc[:, 5], df2.iloc[:, 9]))  # F列到J列的映射
print(f"建立了{len(lookup_dict)}個查找映射")
print(f"查找字典樣本: {dict(list(lookup_dict.items())[:3])}")

# 添加新列
print("\n添加新列...")

# 檢查report1的列數
if df1.shape[1] < 28:
    print(f"警告: Report1列數為{df1.shape[1]}，小於期望的28列，將盡力處理")

# AH列: OrderStat (VLOOKUP功能)
if df1.shape[1] > 6:  # 確保G列存在
    df1['OrderStat'] = df1.iloc[:, 6].map(lookup_dict)  # G列對應的查找
    print("OrderStat列已添加")

    # 顯示OrderStat的值
    print("OrderStat列內容:")
    for i in range(min(len(df1), 5)):
        installat_val = df1.iloc[i, 6] if df1.shape[1] > 6 else "N/A"
        orderstat_val = df1.iloc[i]['OrderStat'] if 'OrderStat' in df1.columns else "N/A"
        print(f"  行{i+1}: Installat='{installat_val}' -> OrderStat='{orderstat_val}'")

# AI列: Remark
def calculate_remark(row):
    try:
        m_col = row.iloc[12] if len(row) > 12 else None  # M列 (索引12)
        ab_col = row.iloc[27] if len(row) > 27 else None  # AB列 (索引27)

        if pd.notna(m_col) and str(m_col).strip() == '01':
            return 'A'
        elif pd.isna(m_col) or str(m_col).strip() == '' or str(m_col).strip() == 'nan':
            if pd.notna(ab_col) and str(ab_col).strip() != '' and str(ab_col).strip() != 'nan':
                return "Can't"
            else:
                return "Out"
        else:
            return "Out"
    except Exception as e:
        print(f"計算Remark時出錯: {e}")
        return "Out"

df1['Remark'] = df1.apply(calculate_remark, axis=1)
print("Remark列已添加")
```

```
# AJ列: O/S days
def calculate_os_days(row):
    try:
        v_col = row.iloc[21] if len(row) > 21 else None  # V列 (索引21)
        if pd.notna(v_col) and str(v_col).strip() != '' and str(v_col).strip() != 'nan':
            try:
                if isinstance(v_col, str):
                    v_date = pd.to_datetime(v_col)
                else:
                    v_date = pd.to_datetime(v_col)
                today_date = pd.Timestamp.now()
                return (today_date - v_date).days + 1
            except:
                return 0
        return 0
    except Exception as e:
        print(f"計算O/S days時出錯: {e}")
        return 0

df1['O/S days'] = df1.apply(calculate_os_days, axis=1)
print("O/S days列已添加")

# AK列: Action Team
def calculate_action_team(row):
    try:
        order_stat = row['OrderStat']
        remark = row['Remark']
        os_days = row['O/S days']

        if pd.notna(order_stat) and str(order_stat).strip() == 'A':
            return 'VEE'
        elif pd.notna(remark) and str(remark).strip() == "Can't":
            return 'eM'
        elif (pd.notna(remark) and str(remark).strip() == 'Out' and
              pd.notna(os_days) and os_days > 6):
            return 'eM'
        else:
            return ''
    except Exception as e:
        print(f"計算Action Team時出錯: {e}")
        return ''

df1['Action Team'] = df1.apply(calculate_action_team, axis=1)
print("Action Team列已添加")

# 只保留OrderStat == 1的行
print("\n篩選數據...")
print(f"篩選前總行數: {len(df1)}")
```

```
# 篩選OrderStat == 1的行
df1_final = df1[df1['OrderStat'] == 1].copy()

print(f"OrderStat為1的行數: {len(df1_final)}")

# 刪除OrderStat列
if 'OrderStat' in df1_final.columns:
    df1_final = df1_final.drop('OrderStat', axis=1)
    print("OrderStat列已刪除")

# 顯示最終結果
print("\n=== 最終結果預覽 ===")
print(f"最終行數: {len(df1_final)}")
if len(df1_final) > 0:
    print(f"列名: {list(df1_final.columns)[:10]}")
    for i in range(min(3, len(df1_final))):
        print(f"第{i+1}行前10列: {df1_final.iloc[i, :10].tolist()}")

# 生成新的文件名（處理後的文件）
processed_filename = f'C:\\Users\\vic25\\R1_report_processed_{today}.xlsx'

# 保存到Excel並設置格式
print(f"\n保存文件到: {processed_filename}")
try:
    with pd.ExcelWriter(processed_filename, engine='openpyxl') as writer:
        df1_final.to_excel(writer, index=False, sheet_name='Sheet1')

        # 獲取工作表以設置格式
        worksheet = writer.sheets['Sheet1']

        # 找到新添加列的位置 (最後3列)
        total_cols = len(df1_final.columns)

        # 設置最後3列（Remark, O/S days, Action Team）為黃色背景
        from openpyxl.styles import PatternFill
        yellow_fill = PatternFill(start_color='FFFF00', end_color='FFFF00', fill_type='solid')

        # 新增的3列是最後3列
        for col in range(total_cols-2, total_cols+1):
            for row in range(1, len(df1_final)+2):
                try:
                    cell = worksheet.cell(row=row, column=col)
                    cell.fill = yellow_fill
                except:
                    pass

    print(f"文件已保存成功")
```

```
    except Exception as e:
        error_msg = f"保存文件時出錯: {str(e)}"
        print(error_msg)
        root = tk.Tk()
        root.withdraw()
        messagebox.showerror("保存錯誤", error_msg)
        root.destroy()
        return None

    # 計算統計信息
    try:
        print("\n計算統計信息...")

        if len(df1_final) > 0:
            vee_count = len(df1_final[df1_final['Action Team'] == 'VEE'])
            em_count = len(df1_final[df1_final['Action Team'] == 'eM'])

            print(f"數據行總數: {len(df1_final)}")
            print(f"VEE案例數: {vee_count}")
            print(f"eM案例數: {em_count}")

            # 顯示Action Team列的所有值進行驗證
            if 'Action Team' in df1_final.columns:
                action_team_values = df1_final['Action Team'].tolist()
                print(f"Action Team所有值: {action_team_values}")
        else:
            vee_count = 0
            em_count = 0
            print("沒有數據行，統計為0")

        # 顯示消息框
        message = f"""處理完成！

原始文件: R1_report_{today}.xlsx
處理後文件: R1_report_processed_{today}.xlsx

VEE follow-up case: {vee_count}
eM follow-up case: {em_count}

文件已保存到: C:\\Users\\vic25\\"""

        # 創建消息框
        root = tk.Tk()
        root.withdraw()
        messagebox.showinfo("處理完成", message)
        root.destroy()
```

```
        print(f"\n處理完成！")
        print(f"VEE follow-up case: {vee_count}")
        print(f"eM follow-up case: {em_count}")

        return processed_filename

    except Exception as e:
        error_msg = f"統計計算時出錯: {str(e)}"
        print(error_msg)
        return processed_filename

def add_excel_formulas(filename):
    """
    在Excel文件中添加公式而不是計算值
    """
    try:
        from openpyxl import load_workbook
        from openpyxl.styles import PatternFill

        wb = load_workbook(filename)
        ws = wb.active

        # 找到列的位置
        headers = [cell.value for cell in ws[1]]

        # 找到新增列的位置
        remark_col = None
        os_days_col = None
        action_team_col = None

        for i, header in enumerate(headers, 1):
            if header == 'Remark':
                remark_col = i
            elif header == 'O/S days':
                os_days_col = i
            elif header == 'Action Team':
                action_team_col = i

        yellow_fill = PatternFill(start_color='FFFF00', end_color='FFFF00', fill_type='solid')

        # 從第2行開始添加公式
        for row in range(2, ws.max_row + 1):
            # Remark公式
            if remark_col:
                formula =
f'=IF(M{row}="01","A",IF(M{row}="",IF(AB{row}<>"","Can\'t","Out"),"Out"))'
                ws.cell(row=row, column=remark_col).value = formula
                ws.cell(row=row, column=remark_col).fill = yellow_fill
```

```
            # O/S days公式
            if os_days_col:
                formula = f'=IF(V{row}="","",TODAY()-V{row}+1)'
                ws.cell(row=row, column=os_days_col).value = formula
                ws.cell(row=row, column=os_days_col).fill = yellow_fill

            # Action Team公式
            if action_team_col and remark_col:
                # 使用列字母而不是數字
                remark_col_letter = ws.cell(row=1, column=remark_col).column_letter
                os_days_col_letter = ws.cell(row=1, column=os_days_col).column_letter

                formula =
f'=IF({remark_col_letter}{row}="A","VEE",IF({remark_col_letter}{row}="Can\'t","eM",IF(AND({r
emark_col_letter}{row}="Out",{os_days_col_letter}{row}>6),"eM","")))'
                ws.cell(row=row, column=action_team_col).value = formula
                ws.cell(row=row, column=action_team_col).fill = yellow_fill

        wb.save(filename)
        print(f"已為 {filename} 添加Excel公式")
        return True

    except Exception as e:
        print(f"添加Excel公式時出錯: {str(e)}")
        return False

# 主程序
if __name__ == "__main__":
    try:
        print("開始處理SAP報告...")
        print(f"處理日期: {datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")

        # 處理報告
        output_file = process_reports()

        if output_file:
            # 詢問是否要添加Excel公式
            root = tk.Tk()
            root.withdraw()

            add_formulas = messagebox.askyesno(
                "添加公式",
                "是否要在Excel文件中添加公式而不是計算值？\n"
                "(這樣其他人可以看到和修改公式)"
            )

            if add_formulas:
```

```
            if add_excel_formulas(output_file):
                messagebox.showinfo("完成", "已添加Excel公式到文件中")
            else:
                messagebox.showwarning("警告", "添加Excel公式時出現問題")

        root.destroy()
        print("程序執行完成！")
    else:
        print("程序執行失敗！")

except Exception as e:
    print(f"程序執行時出現錯誤: {str(e)}")
    root = tk.Tk()
    root.withdraw()
    messagebox.showerror("錯誤", f"程序執行時出現錯誤:\n{str(e)}")
    root.destroy()
```