```
import os
import json
import pandas as pd
import logging
from datetime import datetime
import re
def setup_logging(log_dir):
    os.makedirs(log_dir, exist_ok=True)
    log_file = os.path.join(log_dir, 'json_processor.log')
    logging.basicConfig(filename=log_file, level=logging.INFO,
                        format='%(asctime)s - %(levelname)s - %(message)s')
def sanitize_df_name(name):
    return re.sub(r'\W+', '_', name)
def process_nested_data(data, prefix=''):
    result = {}
    if isinstance(data, dict):
        for key, value in data.items():
            new_key = f"{prefix}{key}"
            if isinstance(value, (dict, list)):
                result.update(process_nested_data(value, f"{new_key}_"))
                result[new_key] = [value]
    elif isinstance(data, list):
        for i, item in enumerate(data):
            new_key = f"{prefix}{i}"
            if isinstance(item, (dict, list)):
                result.update(process_nested_data(item, f"{new_key}_"))
            else:
                result[new_key] = [item]
    return result
def process_json_files(input_dir, base_output_dir):
    # Create timestamped output directory
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    output_dir = os.path.join(base_output_dir, timestamp)
    os.makedirs(output_dir, exist_ok=True)
    # Setup logging in the new output directory
    setup_logging(output_dir)
    logging.info(f"Starting JSON processing from directory: {input_dir}")
    logging.info(f"Output directory created: {output_dir}")
    dfMain = pd.DataFrame()
    nested_dfs = {}
    try:
        for filename in os.listdir(input_dir):
            if filename.endswith('.json'):
                fkID = os.path.splitext(filename)[0]
                file_path = os.path.join(input_dir, filename)
                logging.info(f"Processing file: {filename}")
                try:
                    with open(file_path, 'r') as file:
                        data = json.load(file)
                    # Process first level simple/single value fields
                    main_data = {k: [v] for k, v in data.items() if not isinstance(v, (dict, lister))
                    main_data['fkID'] = [fkID]
                    dfMain = pd.concat([dfMain, pd.DataFrame(main_data)], ignore_index=True)
                    # Process nested data
                    for key, value in data.items():
                        if isinstance(value, (dict, list)):
                            df_name = sanitize_df_name(key)
                            nested_data = process_nested_data(value)
                            nested_data['fkID'] = [fkID]
```

```
if df_name not in nested_dfs:
                                nested_dfs[df_name] = pd.DataFrame()
                            nested_dfs[df_name] = pd.concat([nested_dfs[df_name], pd.DataFrame()
                            ignore_index=True)
                    logging.info(f"Successfully processed file: {filename}")
                except Exception as e:
                    logging.error(f"Error processing file {filename}: {str(e)}")
        # Save all dataframes
        dfMain.to_csv(os.path.join(output_dir, 'dfMain.csv'), index=False)
        logging.info("Saved dfMain.csv")
        for df_name, df in nested_dfs.items():
            df.to_csv(os.path.join(output_dir, f'{df_name}.csv'), index=False)
            logging.info(f"Saved {df_name}.csv")
        logging.info("JSON processing completed successfully")
   except Exception as e:
        logging.error(f"An error occurred during processing: {str(e)}")
if __name__ == "__main__":
    input_directory = input("Enter the input directory path: ")
   base_output_directory = input("Enter the base output directory path: ")
   process_json_files(input_directory, base_output_directory)
   print(f"Processing complete. Check the log file in the output directory for details.")
```