```python
import spikeinterface.full as si
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import subprocess
import math
import os
import re
import scipy
from pathlib import Path
from utils import parse_arguments, load_config, expand_config
#!/usr/bin/env python3

"""
sorting.py

Description:
    This script performs sorting operations.

Usage:
    python sorting.py

Author:
    Your Name

Date:
    YYYY-MM-DD
"""


def main(args):
    # Your main code here
    print("Sorting script")

    # Load configuration from JSON file
    a = load_config(args.config_file)
    s = expand_config(a)

    preprocessed_recordings = {}

    # Filter to only include directories with numeric names (0, 1, 2, 3)
    numeric_dirs = [d for d in os.listdir(s["preprocessing_output_folder"])
                    if os.path.isdir(s["preprocessing_output_folder"] / d) and d.isdigit()]

    print(f"Found numeric directories: {numeric_dirs}")

    for dir_name in numeric_dirs:
        group = int(dir_name)

        recording_path = s["preprocessing_output_folder"] / dir_name
        print(f"Loading recording from: {recording_path}")

        try:
            # Try to load the recording using the SpikeInterface loader
            preprocessed_recordings[group] = si.load_extractor(recording_path)
            print(f"Successfully loaded recording for group {group}")
        except Exception as e:
            print(f"Error loading recording for group {group}: {e}")
            try:
                # Fallback to regular load
                preprocessed_recordings[group] = si.load(recording_path)
                print(f"Successfully loaded recording using alternative method for group {group}")
            except Exception as e2:
                print(f"All loading methods failed for group {group}: {e2}")
                continue

    if not preprocessed_recordings:
```

```
        print("No recordings could be loaded. Exiting.")
        return

    print(f"Successfully loaded {len(preprocessed_recordings)} recordings")

    #docke Get default parameters
    #params = si.get_default_sorter_params(sorter_name_or_class='kilosort4')
    params = {}

    # Performing CAR again is redundant
    params['do_CAR'] = False

    # We do not want to smooth the motion correct in time
    # as we expect some discontinuities between concatenated
    # trigger files
    params['drift_smoothing'] = [0.5, 0.5, 0.5] # Need to confirm that this is a sensible thing to do

    si.set_global_job_kwargs(n_jobs=20, chunk_duration="10s", progress_bar=True)

    sortings = {}
    for group, sub_recording in preprocessed_recordings.items():
        print(f"Processing group {group}")

        # Calculate the number of blocks to use for each group
        # (The recommendation for an entire shank is 5 blocks,
        # so we scale accordingly)
        params['nblocks'] = math.ceil(5 *
            sub_recording.get_probe().get_contact_count() / 384)

        # Define output folder
        ks_sub_folder = s["ks_output_folder"] / f"{group}"
        print(f"Output folder: {ks_sub_folder}")

        # Run the sorter
        sorting = si.run_sorter(
            sorter_name='spikeinterface/kilosort4-base:4.0.18_cuda-12.0.0',
            recording=sub_recording,
            folder=ks_sub_folder,
            singularity_image=True, # Update as necessary
            verbose=True,
            remove_existing_folder=True,
            **params
            )

        # Add the result to the dictionary
        sortings[group] = sorting
        print(f"Sorting complete for group {group}")

    print("All sorting operations completed")

if __name__ == "__main__":
    args = parse_arguments()
    main(args)
```