```
clear all;
[ALLEEG, EEG, CURRENTSET, ALLCOM] = eeglab;

% Define paths
data_path = 'C:\Users\ASUS\Desktop\IITK\6th
Sem\BSE662\Project_Anxiety_prediction(IoP)\EEG_Data\eegdata_all';
Behavior_path = 'C:\Users\ASUS\Desktop\IITK\6th
Sem\BSE662\Project_Anxiety_prediction(IoP)\EEG_Data\behavioural_data_all';
save_path = 'C:\Users\ASUS\Desktop\IITK\6th
Sem\BSE662\Project_Anxiety_prediction(IoP)\EEG_Data\behavioural_data_all\e2_preprocesse
d';

% Loop over all datasets
for i = 1:54
    % Define file names
    filename = sprintf('p%d_import.set', i);
    behavior_file = sprintf('data_env1_p%d_001_.csv', i);
    save_filename = sprintf('p%d_e1_preprocessed.set', i);

    % Load the EEG dataset
    EEG = pop_loadset('filename', fullfile(data_path, filename));
    [ALLEEG, EEG, CURRENTSET] = eeg_store(ALLEEG, EEG, 0);
    eeglab redraw;

    % Channels List Before
    disp({EEG.chanlocs.labels});

    % Define file paths for channel locations
    ced_file = 'C:\Users\ASUS\Documents\live_amp_32.ced';

    % Load Channel labels
    EEG = pop_chanedit(EEG, 'load', {ced_file, 'filetype', 'chanedit'});

    % Channels List After Labelling
    disp({EEG.chanlocs.labels});

    % Channel removal
    EEG = pop_select(EEG, 'rmchannel', {'FT10', 'FT9'});

    % Add FCz as a reference channel
    EEG = pop_chanedit(EEG, 'append', 31, 'changefield', {31, 'labels', 'FCz'});

    % Perform re-referencing using TP9 & TP10 and set FCz as the reference
    EEG = pop_reref(EEG, [29, 30], 'keepref', 'on', 'refloc', struct('labels', 'FCz', 'type', 'EEG', ...
```

```
                                    'theta', 0.7867, 'radius', 0.095376, ...
                                    'X', 27.39, 'Y', -0.3761, 'Z', 88.668, ...
                                    'sph_theta', -0.7867, 'sph_phi', 72.8323, ...
                                    'sph_radius', 92.8028, 'urchan', 34, ...
                                    'ref', 'FCz', 'datachan', 1));

% Removing bad channels
EEG = pop_select(EEG, 'rmchannel', {'TP10', 'TP9'});

% Verify the current reference
if isfield(EEG, 'ref')
   disp(['Current Reference: ' EEG.ref]);
else
   disp('Current Reference field not found. Check manually.');
end

% Apply notch filter
EEG = pop_eegfiltnew(EEG, 49, 51, [], 1, [], 0);

% Apply a high-pass filter at 0.5 Hz
EEG = pop_eegfiltnew(EEG, 'locutoff', 0.5);

% Apply a low-pass filter at 49 Hz
EEG = pop_eegfiltnew(EEG, 'hicutoff', 49);

% Store and update the dataset
[ALLEEG, EEG, CURRENTSET] = eeg_store(ALLEEG, EEG, CURRENTSET);
eeglab redraw;

% Event types
disp(unique({EEG.event.type}));

% Load behavioral data
behavior_data = readtable(fullfile(Behavior_path, behavior_file));

% Find all 'c' events
c_event_idx = find(strcmp({EEG.event.type}, 'c'));

% Ensure we don't exceed the number of available decisions
num_decisions = height(behavior_data);
num_events = length(c_event_idx);
num_to_assign = min(num_decisions, num_events);

% Assign decisions to 'c' events
```

```
    for j = 1:num_to_assign
        EEG.event(c_event_idx(j)).type = char(behavior_data.Decision(j));
    end


    % Epoch extraction (-1 to 1)
    EEG = pop_epoch(EEG, {'stay', 'leave'}, [-1 0.1], 'newname', 'epoched_data', 'epochinfo',
'yes');
    % remove three more channels
    EEG = pop_select(EEG, 'rmchannel', {'T8','T7', 'FC1'});

    % Save preprocessed dataset
    pop_saveset(EEG, 'filename', save_filename, 'filepath', save_path);

    fprintf('Processing completed for Subject %d\n', i);
end

disp('All subjects processed successfully!');
```

PLOTS

```
 % Compute and Plot Spectral Power
    figure;
    [specdata, freqs] = spectopo(EEG.data, 0, EEG.srate, 'plot', 'on');
    xlabel('Frequency (Hz)');
    ylabel('Power (dB)');
    title(sprintf('Spectral Power - Participant%d', i));

    % Compute Spectral Power for Specific Bands
    bands = {'Delta', 'Theta', 'Alpha', 'Beta', 'Gamma'};
    band_ranges = [1 4; 4 8; 8 12; 12 30; 30 50];

    band_powers = zeros(1, length(bands));

    for b = 1:length(bands)
        % Find indices for the band range
        freq_idx = freqs >= band_ranges(b,1) & freqs <= band_ranges(b,2);

        % Compute mean power in this range
        band_powers(b) = mean(specdata(freq_idx));
    end

    % Plot bar graph of band powers
```

```
figure;
bar(band_powers);
set(gca, 'xticklabel', bands);
xlabel('Frequency Band');
ylabel('Mean Power (dB)');
title(sprintf('Spectral Band Power - Participant %d', i));
```