```matlab
%% ================== 1️⃣Load & Preprocess Data ==================
clc; clear; close all;

% Load dataset
opts = detectImportOptions('cs.csv');
opts = setvaropts(opts, 'DateTimeColumn', 'InputFormat', 'MM/dd/uuuu HH:mm:ss'); %
Set Date Format
dataTable = readtable('your_dataset.csv', opts);

% Convert DateTime to numerical format (if needed)
dataTable.DateTimeColumn = datenum(dataTable.DateTimeColumn);

% Convert table to array
X = table2array(dataTable(:, 2:end));  % Assuming DateTime is first column
disp("✅Data Successfully Loaded!");

% Check for missing values and fill
X = fillmissing(X, 'linear');
disp("✅Missing Values Handled!");

% Display size
disp("Initial Size of X:"); disp(size(X));

%% ================== 2️⃣Normalize Data ==================
[X, X_min, X_max] = normalizeData(X);
disp("✅Data Normalized!");

%% ================== 3️⃣Prepare Training Data ==================
% Define Training & Target Variables
X_train = X(1:end-10, :); % Training features (excluding last 10)
Y_train = X(2:end-9, :); % Target values (next time step)

% Ensure Data is Not Empty
if isempty(X_train) || isempty(Y_train)
    error("⍰ Error: No valid data available for training. Check preprocessing.");
end
disp("✅Training Data Prepared!");

%% ================== 4️⃣Check GPU Availability ==================
try
    gpuInfo = gpuDevice();
    executionEnv = 'gpu';
    disp("✅Using GPU for training.");
catch
    executionEnv = 'cpu';
    disp("□ GPU not available. Using CPU (training may be slower).");
end

%% ================== 5️⃣Define & Train LSTM Model ==================
layers = [
    sequenceInputLayer(size(X_train, 2))
    lstmLayer(50, 'OutputMode', 'sequence')
    fullyConnectedLayer(size(Y_train, 2))
    regressionLayer
];

options = trainingOptions('adam', ...
    'MaxEpochs', 25, ...
    'MiniBatchSize', 128, ...
    'Shuffle', 'every-epoch', ...
    'ExecutionEnvironment', executionEnv, ...
    'Plots', 'training-progress');

disp("⏳ Training Model...");
net = trainNetwork(X_train', Y_train', layers, options);
disp("✅Model Training Completed!");

%% ================== 6️⃣Make Predictions ==================
X_test = X(end-9:end, :);
Y_pred = predict(net, X_test')';

% Denormalize predictions
Y_pred = denormalizeData(Y_pred, X_min, X_max);
disp("✅Forecasting Completed Successfully!");

%% ================== 7️⃣Plot Predictions ==================
figure;
t = datetime(dataTable.DateTimeColumn(end-9:end), 'ConvertFrom', 'datenum');

subplot(3,1,1);
plot(t, Y_pred(:,1), 'r'); title('Predicted Solar Power'); grid on;

subplot(3,1,2);
plot(t, Y_pred(:,2), 'b'); title('Predicted Battery SOC'); grid on;

subplot(3,1,3);
plot(t, Y_pred(:,3), 'g'); title('Predicted Load Demand'); grid on;

disp("✅Results Plotted!");

%% ================== ▯ Helper Functions ==================
function [X_norm, X_min, X_max] = normalizeData(X)
    X_min = min(X);
    X_max = max(X);
    X_norm = (X - X_min) ./ (X_max - X_min);
end

function X_denorm = denormalizeData(X_norm, X_min, X_max)
    X_denorm = X_norm .* (X_max - X_min) + X_min;
end
```