/\* Note: This script assumes each student has exactly one first name and exactly one last name and zero or more middle names. If a student has more than one first name AND both first names are recorded in the Google sheet, or if a student has more than one last name AND both last names are recorded in the Google sheet, then this script will not work as-is.

One solution is to put student full names in a single column in the spreadsheet in the form "First First Middle Middle Last Last" and adjust the findStudentRow() function so it searches for the full name instead of the last and then the first. In that case, csv files with names of the form (first first middle middle last last) will be readable as-is, but names of the form (last last, first first middle middle) will need reordered. Split at the comma-space, and concatenate the second part with a space with the first part. Note that even this fix will not work if a student actually has a comma in their name.

\*/

// CHANGE AT THE BEGINNING OF EACH SEMESTER: the following variables, and the list of column name matches.

var RECORD\_SHEET\_TITLE = "Record"; // The name of the main grades sheet

var RECORD\_COLUMN\_TITLES\_RANGE = "K3:3"; // Where the column titles are in the Record sheet

var RECORD\_NAMES\_RANGE = "B4:C33"; // Where the student names are in the Record sheet

var ENGAGEMENT\_SHEET\_TITLE = "Engagement"; // The name of the engagement tracking sheet

var ENGAGEMENT\_COLUMN\_TITLES\_RANGE = "D3:3"; // Where the column titles are in the Record sheet

var ENGAGEMENT\_NAMES\_RANGE = "B4:C33"; // Where the student names are in the Engagement sheet

var TIME\_SHEET\_TITLE = "Time"; // The name of the time tracking sheet

var TIME\_COLUMN\_TITLES\_RANGE = "D3:3"; // Where the column titles are in the Record sheet

var TIME\_NAMES\_RANGE = "B4:C33"; // Where the student names are in the Time sheet

var CSV\_ENGAGEMENT\_COL\_INDEX = 9; // The column index where students reported their engagement in the quiz report csv

var CSV\_TIME\_COL\_INDEX = 13; // The column index where students reported their time spent in the quiz report csv

var CSV\_GRADES\_NAMES\_FIRST\_INDEX = 3; // The index of the row where the first student name appears

var CSV\_QUIZ\_NAMES\_FIRST\_INDEX = 1; // The index of the row where the first student name appears

var IGNORED\_STUDENTS\_COMMA = ["Student, Test", ]; // The names of students to ignore as they appear in the grades CSV

var IGNORED\_STUDENTS\_NOCOMMA = ["Test Student", ]; // The names of students to ignore as they appear in the quiz report CSV

// When importGrades is called, each string in this array will be replaced with the index of the matching column. The first index of each row is the name of the column in the grades csv, and the second index of each row is the name of the column in the Records spreadsheet.

var csvColToSheetColGrades = [

["1 -- Function Evaluation (8380225)", "1-FE"],

["1 -- Function Basics (8380223)", "1-FB"],

["1 -- Transformations of Functions (8380233)", "1-TF"],

["1 -- Solving Linear Systems (8380231)", "1-SLS"],

["1 -- Definition of Logarithm (8380219)", "1-DL"],

["1 -- Evaluate Exponential and Logarithmic Functions (8380221)", "1-EELF"],

["1 -- Properties of Logarithms (8380227)", "1-PL"],

["1 -- Solve Logarithmic Equations and IVT (8380229)", "1-SLE"],

["2 -- Derivative Rules Power Function Sum Difference Constant Multiple (8380237)", "2-DR(basic)"],

["2 -- Derivative Rules Product and Quotient (8380239)", "2-DR(pq)"],

["2 -- Derivative Rules Function Composition - the Chain Rule (8380235)", "2-DR(chain)"],

["2 -- Linearization (8380245)", "2-L"],

["2 -- Partial Derivatives (8380247)", "2-PD"],

["2 -- First Derivative and Extrema (8380241)", "2-FDE"],

["2 -- Second Derivative and Extrema (8380249)", "2-SDE"],

["2 -- Least Squares Prediction and Extrapolation (8380243)", "2-LSq"],

["3 -- Probability Basics (8380263)", "3-P(basic)"],

["3 -- Discrete Random Variables and Expected Value (8380261)", "3-DRV"],

["3 -- Riemann Sums & Definite Integrals (8380265)", "3-RS"],

["3 -- Compute Probability Continuous Random Variables (8380257)", "3-CRV"],

["3 -- Definite Integrals (8380259)", "3-DI"],

["3 -- Calculate Parameters of Continuous Random Variables Identifying Expected Value and Variance (8380253)", "3-EVV"],

["3 -- CDFs PDFs and the Fundamental Theorem of Calculus (8380255)", "3-CDFpdf"],

["3 -- Antiderivatives or Indefinite Integrals (8380251)", "3-II"],

["Project 1 Task 1 - Cohesive Narrative (8380301)", "P1T1"],

["Project 1 Task 2 - Cohesive Narrative (8380303)", "P1T2"],

["Project 1 Task 3 - Cohesive Narrative (8380305)", "P1T3"],

["Project 1 Task 4 - Cohesive Narrative (8380307)", "P1T4"],

["Project 2 Task 1 - Cohesive Narrative (8380311)", "P2T1"],

["Project 2 Task 2 - Cohesive Narrative (8380313)", "P2T2"],

["Project 2 Task 3 - Cohesive Narrative (8380315)", "P2T3"],

["Project 3 Task 1 - Cohesive Narrative (8380319)", "P3T1"],

["Project 3 Task 2 - Cohesive Narrative (8380321)", "P3T2"],

["Project 1 (8380299)", "P1"],

["Project 2 (8380309)", "P2"],

["Project 3 (8380317)", "P3"],

["Roll Call Attendance (8381519)", "Attendance"]

];

// Add functions to the menu bar.

//@OnlyCurrentDoc

function onOpen(e) {

var ui = SpreadsheetApp.getUi();

ui.createMenu("Import CSV data 👉️")

.addItem("Import Grade Info", 'importGrades')

.addItem("Import Weekly Report", 'importWeeklyReport')

.addToUi();

}

function findStudentRow(sheetRange, fullName){

var sheetArray = sheetRange.getValues();

var rowOffset = sheetRange.getRow();

var lastName = "";

var firstName = "";

if (fullName.indexOf(",") == -1){

// The name is in this order: "first middle last"

fullName = fullName.split(' ');

lastName = fullName[fullName.length - 1];

firstName = fullName[0];

}else{

// The name is in this order: "last, first middle"

fullName = fullName.replace(",", "").split(' ');

lastName = fullName[0];

firstName = fullName[1];

}

var firstNames = sheetArray.map(x => x[1]);

var lastNames = sheetArray.map(x => x[0]);

// Find the student in the Records sheet

var lastNameRow = lastNames.indexOf(lastName);

if (lastNameRow == -1){

return -1;

}

var nameRow = firstNames.indexOf(firstName, lastNameRow);

if (nameRow == -1){

return -1;

}

return nameRow + rowOffset;

}

// Replaces the string names in the above 2D array with their indexes.

function replaceColTitlesWithIndicesGrades(sheet, csvArray){

var sheetColTitles = sheet.getRange(RECORD\_COLUMN\_TITLES\_RANGE);

var sheetColOffset = sheetColTitles.getColumn();

sheetColTitles = sheetColTitles.getValues()[0];

// Go through each row in the column converter

for (var iTitles = 0; iTitles < csvColToSheetColGrades.length; iTitles++){

// Get the csv column title we're looking for.

var csvTitle = csvColToSheetColGrades[iTitles][0];

// Loop through the columns of csvArray to find it.

var found = false;

for (var iCSV = 0; iCSV < csvArray[0].length; iCSV++){

// If we find it, then update the value in the column converter.

if (csvArray[0][iCSV] == csvTitle){

csvColToSheetColGrades[iTitles][0] = iCSV;

found = true;

break;

}

}

// If it wasn't found, we need to quit

if (!found){

displayToastAlert("Aborting: Column not found in csv '" + csvName + "'");

return false;

}

// Get the Record sheet’s column title we're looking for.

var recordTitle = csvColToSheetColGrades[iTitles][1];

// Loop through the columns of Records to find it.

found = false;

for (var iRecords = 0; iRecords < sheetColTitles.length; iRecords++){

// If we find it, then update the value in the column converter.

if (sheetColTitles[iRecords] == recordTitle){

// Need to add the offset, because the first item in sheetColTitles is not the item in the first column.

csvColToSheetColGrades[iTitles][1] = iRecords + sheetColOffset;

found = true;

break;

}

}

// If it wasn't found, we need to quit

if (!found){

displayToastAlert("Aborting: Column not found in Record '" + recordsName + "'");

return false;

}

}

// Everything succeeded, so let's say so!

return true;

}

// Goes through the CSV of grades and adds them all to the Records spreadsheet

// For row in contents' name column

// Get first name and last name

// Find the row of the last name in master sheet

// From there, find the row of the first name in master sheet

// For column in important columns of the master sheet

// Find matching column in contents

// Get the data for contents' row and column

// Paste it into master sheet's row and column

function importGrades() {

// Get the contents of the CSV

var csvContents = importCSVFromDrive()[0];

// Get the Records sheet

var ss = SpreadsheetApp.getActiveSpreadsheet();

var recordSheet = ss.getSheetByName(RECORD\_SHEET\_TITLE);

// Try to find the names of each column and replace it with its index

// If this returns false, it failed and we need to quit.

if (!replaceColTitlesWithIndicesGrades(recordSheet, csvContents)){

return;

}

// Get the range that contains the student names

var sheetNamesRange = recordSheet.getRange(RECORD\_NAMES\_RANGE);

// Go through each name in the csv

for (let csvNameIndex = CSV\_GRADES\_NAMES\_FIRST\_INDEX; csvNameIndex < csvContents.length; csvNameIndex++){

// Get the name of the student

var fullName = csvContents[csvNameIndex][0];

// Ignore some students

var shouldIgnore = false;

for (ignoreName of IGNORED\_STUDENTS\_COMMA){

if (fullName == ignoreName){

shouldIgnore = true;

break;

}

}

if (shouldIgnore){

continue;

}

var sheetStudentRow = findStudentRow(sheetNamesRange, fullName);

if (sheetStudentRow == -1){

displayToastAlert("Name not found in Records: " + fullName);

return;

}

// Loop through each important column in the csvColToSheetColGrades

for (var colConverterIndex = 0; colConverterIndex < csvColToSheetColGrades.length; colConverterIndex ++){

// and replace the value in Records with the value in the csv

var csvCol = csvColToSheetColGrades[colConverterIndex][0];

var sheetCol = csvColToSheetColGrades[colConverterIndex][1];

recordSheet.getRange(sheetStudentRow, sheetCol).setValue(csvContents[csvNameIndex][csvCol]);

}

}

}

// Goes through the CSV of the quiz responses and adds them all to the Time and Engagement spreadsheets

// Get the name of the file

// For row in contents' name column

// Get first name and last name

// Find the row of the last name in Time and in Engagement

// From there, find the row of the first name in Time and Engagement

// Find engagement column in contents

// Get the data for the student in contents

// Paste it into Engagement's correct cell

// Find hours column in contents

// Get the data for the student in contents

// Paste it into Time's correct cell

function importWeeklyReport(){

var tup = importCSVFromDrive("\nIt must be named after the week it is for (W08.csv)");

var csvContents = tup[0];

var csvFileName = tup[1];

// Get important ranges in the spreadsheet

var ss = SpreadsheetApp.getActiveSpreadsheet();

var timeSheet = ss.getSheetByName(TIME\_SHEET\_TITLE);

var engagementSheet = ss.getSheetByName(ENGAGEMENT\_SHEET\_TITLE);

// Get the ranges where the names of the students are

var timeNamesRange = timeSheet.getRange(TIME\_NAMES\_RANGE);

var engagementNamesRange = engagementSheet.getRange(ENGAGEMENT\_NAMES\_RANGE);

// Turn the column names into a 1d array

var timeColRange = timeSheet.getRange(TIME\_COLUMN\_TITLES\_RANGE);

var timeColTitles = timeColRange.getValues()[0];

var timeColOffset = timeColRange.getColumn();

var engagementColRange = engagementSheet.getRange(ENGAGEMENT\_COLUMN\_TITLES\_RANGE);

var engagementColTitles = engagementColRange.getValues()[0];

var engagementColOffset = engagementColRange.getColumn();

// Go through each name in the csv

for (let csvNameIndex = CSV\_QUIZ\_NAMES\_FIRST\_INDEX; csvNameIndex < csvContents.length; csvNameIndex++){

// Get the name of the student

var fullName = csvContents[csvNameIndex][0];

// Ignore some students

var shouldIgnore = false;

for (ignoreName of IGNORED\_STUDENTS\_NOCOMMA){

if (fullName == ignoreName){

shouldIgnore = true;

break;

}

}

if (shouldIgnore){

continue;

}

var timeStudentRow = findStudentRow(timeNamesRange, fullName);

if (timeStudentRow == -1){

displayToastAlert("Name not found in Time: " + fullName);

return;

}

// Find the student in the Engagement sheet

var engagementStudentRow = findStudentRow(engagementNamesRange, fullName);

if (engagementStudentRow == -1){

displayToastAlert("Name not found in Engagement: " + fullName);

return;

}

// name is the filename, so the first three characters are in the form W00

// and that's exactly the column we want.

var timeColTitle = csvFileName.slice(0, 3);

var timeCol = timeColTitles.indexOf(timeColTitle);

if (timeCol == -1){

displayToastAlert("Column not found in Time: " + timeColTitle);

return;

}

timeCol += timeColOffset;

// The engagement column just needs 'E-' on the front of that.

var engagementColTitle = "E-" + timeColTitle;

var engagementCol = engagementColTitles.indexOf(engagementColTitle);

if (engagementCol == -1){

displayToastAlert("Column not found in Engagement: " + engagementColTitle);

return;

}

engagementCol += engagementColOffset;

// Copy/paste the data

timeSheet.getRange(timeStudentRow, timeCol).setValue(csvContents[csvNameIndex][CSV\_TIME\_COL\_INDEX]);

engagementSheet.getRange(engagementStudentRow, engagementCol).setValue(csvContents[csvNameIndex][CSV\_ENGAGEMENT\_COL\_INDEX]);

}

}

// Displays a text box for user for input.

function promptUserForInput(promptText) {

var ui = SpreadsheetApp.getUi();

var prompt = ui.prompt(promptText);

var response = prompt.getResponseText();

return response;

}

//Displays an alert as a Toast message

function displayToastAlert(message) {

SpreadsheetApp.getActive().toast(message, "⚠️ Alert");

}

//Function to import CSV files from Google Drive

function importCSVFromDrive(prompt = "") {

var fileName = promptUserForInput("Please enter the name of the CSV file to import from Google Drive:" + prompt);

var files = findFilesInDrive(fileName);

if(files.length === 0) {

displayToastAlert("No files with name \"" + fileName + "\" were found in Google Drive.");

return;

} else if(files.length > 1) {

displayToastAlert("Multiple files with name " + fileName +" were found. This program does not support picking the right file yet.");

return;

}

var file = files[0];

var things = file.getBlob();

var more = things.getDataAsString();

var contents = parseCSV(more);

//var contents = Utilities.parseCsv(file.getBlob().getDataAsString());

return [contents, fileName];

}

// THIS FUNCTION FROM https://stackoverflow.com/a/14991797

// Necessary because the weekly reports have newlines and user input

// that needs sanitized.

function parseCSV(str) {

var arr = [];

var quote = false; // 'true' means we're inside a quoted field

// Iterate over each character, keep track of current row and column (of the returned array)

for (var row = 0, col = 0, c = 0; c < str.length; c++) {

var cc = str[c], nc = str[c+1]; // Current character, next character

arr[row] = arr[row] || []; // Create a new row if necessary

arr[row][col] = arr[row][col] || ''; // Create a new column (start with empty string) if necessary

// If the current character is a quotation mark, and we're inside a

// quoted field, and the next character is also a quotation mark,

// add a quotation mark to the current column and skip the next character

if (cc == '"' && quote && nc == '"') { arr[row][col] += cc; ++c; continue; }

// If it's just one quotation mark, begin/end quoted field

if (cc == '"') { quote = !quote; continue; }

// If it's a comma and we're not in a quoted field, move on to the next column

if (cc == ',' && !quote) { ++col; continue; }

// If it's a newline (CRLF) and we're not in a quoted field, skip the next character

// and move on to the next row and move to column 0 of that new row

if (cc == '\r' && nc == '\n' && !quote) { ++row; col = 0; ++c; continue; }

// If it's a newline (LF or CR) and we're not in a quoted field,

// move on to the next row and move to column 0 of that new row

if (cc == '\n' && !quote) { ++row; col = 0; continue; }

if (cc == '\r' && !quote) { ++row; col = 0; continue; }

// Otherwise, append the current character to the current column

arr[row][col] += cc;

}

return arr;

}

//Returns files in Google Drive that have a certain name.

function findFilesInDrive(filename) {

var files = DriveApp.getFilesByName(filename);

var result = [];

while(files.hasNext())

result.push(files.next());

return result;

}