// === WholesalingLeadEngine.gs === //

function onEdit(e) {

const sheet = e.source.getActiveSheet();

const row = e.range.getRow();

if (sheet.getName() !== "Master Leads Database" || row === 1) return;

const headers = sheet.getRange(1, 1, 1, sheet.getLastColumn()).getValues()[0];

const data = sheet.getRange(row, 1, 1, headers.length).getValues()[0];

// === Assign Row ID if missing ===

const rowIdIndex = headers.indexOf("Row ID");

if (!data[rowIdIndex]) {

sheet.getRange(row, rowIdIndex + 1).setValue("RID-" + new Date().getTime());

}

// === Auto Motivation Score ===

const conditionIndex = headers.indexOf("Condition Notes");

const timelineIndex = headers.indexOf("Timeline");

const motivationIndex = headers.indexOf("Motivation Score");

let score = 0;

if (data[conditionIndex]?.toLowerCase().includes("vacant") || data[conditionIndex]?.toLowerCase().includes("major repair")) score += 2;

if (data[timelineIndex]?.toLowerCase().includes("asap") || data[timelineIndex]?.toLowerCase().includes("30 days")) score += 2;

sheet.getRange(row, motivationIndex + 1).setValue(score);

// === Trigger Lauren AI Bot Logic ===

const statusIndex = headers.indexOf("Lead Status");

const crmStageIndex = headers.indexOf("CRM Stage");

if (data[statusIndex] === "No Response" && score >= 2) {

sheet.getRange(row, crmStageIndex + 1).setValue("Re-engagement Sent");

SpreadsheetApp.getUi().alert("Lauren Bot: Re-engagement triggered due to high motivation.");

}

// === Flag Duplicates by Property Address ===

const addressIndex = headers.indexOf("Property Address");

const allAddresses = sheet.getRange(2, addressIndex + 1, sheet.getLastRow() - 1).getValues().flat();

const currentAddress = data[addressIndex];

const duplicateCount = allAddresses.filter(addr => addr === currentAddress).length;

if (duplicateCount > 1) {

SpreadsheetApp.getUi().alert("Duplicate property address detected: " + currentAddress);

}

}

// === Custom Menu Hook === //

function onOpen() {

const ui = SpreadsheetApp.getUi();

ui.createMenu("Lead Engine")

.addItem("Transfer from Gateway", "transferFromPropStreamGateway")

.addItem("Run Skiptrace on All ❌ Rows", "triggerSkiptrace") // ✅ New item 052925

.addItem("Export Enriched Leads to CSV", "exportEnrichedLeadsToCSV")

.addItem("Score Motivation Manually", "manualScoreMotivation")

.addItem("Push to Taskify CRM", "pushToTaskify")

.addItem("Restore Selected Leads to Master", "restoreSelectedLeadsToMaster")

.addItem("Check Header Consistency", "checkHeaderConsistency") // ✅ New item 052825

.addToUi();

}

function triggerSkiptrace() {

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Master Leads Database");

const data = sheet.getDataRange().getValues();

const headers = data[0];

const statusCol = headers.indexOf("Skiptrace Status");

const firstNameCol = headers.indexOf("Owner 1 First Name");

const lastNameCol = headers.indexOf("Owner 1 Last Name");

const cityCol = headers.indexOf("City");

const stateCol = headers.indexOf("State");

const zipCol = headers.indexOf("Zip");

const urlCol = 69; // Column BQ

const timeCol = 59; // Column BG

// Clear previous skiptrace URLs

sheet.getRange(2, urlCol, sheet.getLastRow() - 1).clearContent();

let count = 0;

const now = Utilities.formatDate(new Date(), Session.getScriptTimeZone(), "MM/dd/yyyy HH:mm:ss");

for (let i = 1; i < data.length; i++) {

const status = data[i][statusCol];

const first = data[i][firstNameCol];

const last = data[i][lastNameCol];

const city = data[i][cityCol];

const state = data[i][stateCol];

const zip = data[i][zipCol];

if ((status === "" || status === "❌") && first && last && city && state && zip && count < 10) {

const url = `https://www.truepeoplesearch.com/results?name=${encodeURIComponent(first + ' ' + last)}&citystatezip=${encodeURIComponent(city + ', ' + state + ' ' + zip)}`;

sheet.getRange(i + 1, urlCol).setValue(url);

sheet.getRange(i + 1, timeCol).setValue(now);

count++;

}

}

SpreadsheetApp.getUi().alert(`✅ Prepared ${count} skiptrace URLs.\nOld values cleared.\nTimestamps set in Column BG.`);

}

function manualScoreMotivation() {

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Master Leads Database");

const data = sheet.getDataRange().getValues();

const headers = data[0];

for (let i = 1; i < data.length; i++) {

const condition = data[i][headers.indexOf("Condition Notes")];

const timeline = data[i][headers.indexOf("Timeline")];

let score = 0;

if (condition?.toLowerCase().includes("vacant") || condition?.toLowerCase().includes("major repair")) score += 2;

if (timeline?.toLowerCase().includes("asap") || timeline?.toLowerCase().includes("30 days")) score += 2;

sheet.getRange(i + 1, headers.indexOf("Motivation Score") + 1).setValue(score);

}

}

function pushToTaskify() {

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Master Leads Database");

const data = sheet.getDataRange().getValues();

const headers = data[0];

const payloads = [];

for (let i = 1; i < data.length; i++) {

if (data[i][headers.indexOf("CRM Stage")] === "Ready for Offer") {

payloads.push({

name: data[i][headers.indexOf("Owner Name")],

address: data[i][headers.indexOf("Property Address")],

phone: data[i][headers.indexOf("Phone Number")],

email: data[i][headers.indexOf("Email")],

score: data[i][headers.indexOf("Motivation Score")] || 0

});

}

}

Logger.log("Payloads ready to send to CRM:", payloads);

// Integration logic goes here when ready.

}

function restoreSelectedLeadsToMaster() {

const ss = SpreadsheetApp.getActiveSpreadsheet();

const archiveSheet = ss.getSheetByName("PropStream Imports Gateway");

const masterSheet = ss.getSheetByName("Master Leads Database");

const archiveRange = archiveSheet.getActiveRange();

const archiveData = archiveSheet.getDataRange().getValues();

const headers = archiveData[0];

const rowIdIndex = headers.indexOf("Row ID");

if (rowIdIndex === -1) {

SpreadsheetApp.getUi().alert("Missing 'Row ID' column.");

return;

}

const masterRowIds = masterSheet.getRange(2, rowIdIndex + 1, masterSheet.getLastRow() - 1).getValues().flat();

let restoredCount = 0;

archiveRange.getValues().forEach((row, i) => {

const archiveRowIndex = archiveRange.getRow() + i;

const rowId = row[rowIdIndex];

if (!rowId || masterRowIds.includes(rowId)) return; // Skip if blank or already in Master

masterSheet.appendRow(row);

restoredCount++;

});

SpreadsheetApp.getUi().alert(`${restoredCount} lead(s) restored to Master Leads Database.`);

}

function doPost(e) {

const data = JSON.parse(e.postData.contents);

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Master Leads Database");

const leads = sheet.getDataRange().getValues();

const headers = leads[0];

const rowIndex = leads.findIndex((row, i) =>

i > 0 &&

row[headers.indexOf("Full Address")] === data.address &&

row[headers.indexOf("Owner 1 First Name")] === data.firstName &&

row[headers.indexOf("Owner 1 Last Name")] === data.lastName

);

if (rowIndex > 0) {

const row = rowIndex + 1;

sheet.getRange(row, headers.indexOf("Owner 1 Phone") + 1).setValue(data.phone1 || "");

sheet.getRange(row, headers.indexOf("Owner 1 Email") + 1).setValue(data.email1 || "");

sheet.getRange(row, headers.indexOf("Skiptrace Status") + 1).setValue("✅");

}

return ContentService.createTextOutput("Success");

}

// === Optimized transferFromPropStreamGateway === //

function transferFromPropStreamGateway() {

const ss = SpreadsheetApp.getActiveSpreadsheet();

const gateway = ss.getSheetByName("PropStream Imports Gateway");

const master = ss.getSheetByName("Master Leads Database");

const gatewayData = gateway.getDataRange().getValues();

const headers = gatewayData[0];

const rowIdIndex = headers.indexOf("Row ID");

const transferIndex = headers.indexOf("Successfully Transferred");

const dateIndex = headers.indexOf("Transfer Date");

if (rowIdIndex === -1 || transferIndex === -1 || dateIndex === -1) {

SpreadsheetApp.getUi().alert("Missing required columns: Row ID, Successfully Transferred, or Transfer Date");

return;

}

const rowsToTransfer = [];

const rowIdUpdates = [];

const transferUpdates = [];

const dateUpdates = [];

const gatewayUpdateRanges = [];

for (let i = 1; i < gatewayData.length; i++) {

const row = gatewayData[i];

if (row[transferIndex] === "✅") continue;

const newRow = [...row];

const rowId = "RID-" + new Date().getTime() + "-" + i;

newRow[rowIdIndex] = rowId;

rowsToTransfer.push(newRow);

rowIdUpdates.push([rowId]);

transferUpdates.push(["✅"]);

dateUpdates.push([Utilities.formatDate(new Date(), Session.getScriptTimeZone(), "MM/dd/yyyy HH:mm:ss")]);

gatewayUpdateRanges.push(i + 1); // row index for update later

}

if (rowsToTransfer.length === 0) {

SpreadsheetApp.getUi().alert("No new rows to transfer.");

return;

}

// Append to Master Leads Database

const masterStartRow = master.getLastRow() + 1;

master.getRange(masterStartRow, 1, rowsToTransfer.length, rowsToTransfer[0].length).setValues(rowsToTransfer);

// Batch update Gateway sheet

gatewayUpdateRanges.forEach((rowNum, index) => {

gateway.getRange(rowNum, rowIdIndex + 1).setValue(rowIdUpdates[index][0]);

gateway.getRange(rowNum, transferIndex + 1).setValue(transferUpdates[index][0]);

gateway.getRange(rowNum, dateIndex + 1).setValue(dateUpdates[index][0]);

});

SpreadsheetApp.getUi().alert(`${rowsToTransfer.length} row(s) transferred to Master Leads Database.`);

}

// === ExportEnrichedLeadsToCSV.gs === //

function exportEnrichedLeadsToCSV() {

const sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Master Leads Database");

const data = sheet.getDataRange().getValues();

const headers = data[0];

// Remove empty rows

const filteredData = data.filter((row, i) => i === 0 || row.join("").trim() !== "");

// Convert data to CSV format

const csv = filteredData.map(row => row.map(value => `"${value}"`).join(",")).join("\n");

// Create file name with timestamp

const now = new Date();

const timestamp = Utilities.formatDate(now, Session.getScriptTimeZone(), "MMddyy\_HHmmss");

const fileName = `EnrichedLeads\_${timestamp}.csv`;

// Create CSV file in user's Drive

const folderId = "1-DDGBa3z\_OoQe\_Kbf5MXHsYmHw\_106x7";

const folder = DriveApp.getFolderById(folderId);

const file = folder.createFile(fileName, csv, MimeType.CSV);

// Notify user with link

SpreadsheetApp.getUi().alert(`CSV Export Complete! File saved to your Google Drive as: ${fileName}`);

}

// === HeaderConsistencyChecker.gs === //

function checkHeaderConsistency() {

const ss = SpreadsheetApp.getActiveSpreadsheet();

const master = ss.getSheetByName("Master Leads Database");

const gateway = ss.getSheetByName("PropStream Imports Gateway");

if (!master || !gateway) {

SpreadsheetApp.getUi().alert("One or both sheets are missing.");

return;

}

const masterHeaders = master.getRange(1, 1, 1, master.getLastColumn()).getValues()[0];

const gatewayHeaders = gateway.getRange(1, 1, 1, gateway.getLastColumn()).getValues()[0];

const differences = [];

const maxLen = Math.max(masterHeaders.length, gatewayHeaders.length);

for (let i = 0; i < maxLen; i++) {

const m = masterHeaders[i] || "[none]";

const g = gatewayHeaders[i] || "[none]";

if (m !== g) {

differences.push(`Column ${i + 1}: Master = '${m}' | Gateway = '${g}'`);

}

}

if (differences.length === 0) {

SpreadsheetApp.getUi().alert("✅ Headers are consistent across both sheets.");

} else {

const message = "❌ Header differences detected:\n\n" + differences.join("\n");

SpreadsheetApp.getUi().alert(message);

}

}