Homework Assignment – Automating data combination and distribution with google apps script

Report by Riina Kikkas

Spreadsheet link

Scenario Explanation

The goal of this project was to create a simple warehouse management system to support inventory checking. The idea was to collect product data—such as shoe models, sizes, shelf locations, and quantities—from multiple input spreadsheets and combine everything into one centralized Master sheet. After the data was gathered, the next step was to automatically distribute the items into separate spreadsheets based on their shelf codes. This makes it easier to review the stock on each shelf during inventory. In the end, the process helps organize the warehouse data.

Function: makeMaster()

The function gathers shoe data from every spreadsheet stored in the Shoes_input folder and compiles it into a single, clean Master sheet. When the function runs, it opens the main spreadsheet and removes all existing sheets except the one named “Master.” It then clears that sheet and adds a new header row to prepare it for fresh data. After the Master sheet is reset, the function processes each file in the input folder by opening the first sheet of every spreadsheet, reading all rows, and skipping the header. Each valid data row—containing the product, size, shelf, and quantity—is added to the Master sheet. By the end of the process, the Master sheet contains a complete and organized list of all shoe entries from all input files, ready for inventory or further distribution.

```
/**************************************
 * TASK 1 — Combine all shoe files into Master Sheet
 **************************************/
function makeMaster() {

  // 1) INPUT folder ID (Adidas, Puma, Nike, Reebok files)
  let INPUT_FOLDER_ID = "1aOMBRxckZmS-RF-LlLenve32pl2GdpDg";

  let inputFolder = DriveApp.getFolderById(INPUT_FOLDER_ID);
  let masterBook = SpreadsheetApp.openByUrl(
    "https://docs.google.com/spreadsheets/d/1xRMuz8L6UDDV5-S2rLEbRz7x8dxS2GM6ISxtJa_8Fzo/edit"
  );

  // Get or create Master sheet
  let masterSheet = masterBook.getSheetByName("Master");
  if (!masterSheet) {
    masterSheet = masterBook.insertSheet("Master");
  }

  // Delete all other sheets, keep only "Master"
  let sheets = masterBook.getSheets();
  for (let i = 0; i < sheets.length; i++) {
    if (sheets[i].getName() !== "Master") {
      masterBook.deleteSheet(sheets[i]);
    }
  }

  // Clear Master and add header
  masterSheet.clear();
  masterSheet.appendRow(["product", "size", "shelf", "quantity"]);

```

```
  // Loop through all files in the input folder
  for (let files = inputFolder.getFiles(); files.hasNext();) {
    let file = files.next();
    let ss = SpreadsheetApp.openById(file.getId());
    let sheet = ss.getSheets()[0];
    let data = sheet.getDataRange().getValues();

    // Skip header row, start from row index 1
    for (let i = 1; i < data.length; i++) {
      let row = data[i];

      // Skip the row if it is completely empty
      if (row.join("").trim() === "") {
        continue;
      }

      // Add row to Master sheet
      masterSheet.appendRow(row);
    }
  }
}

```

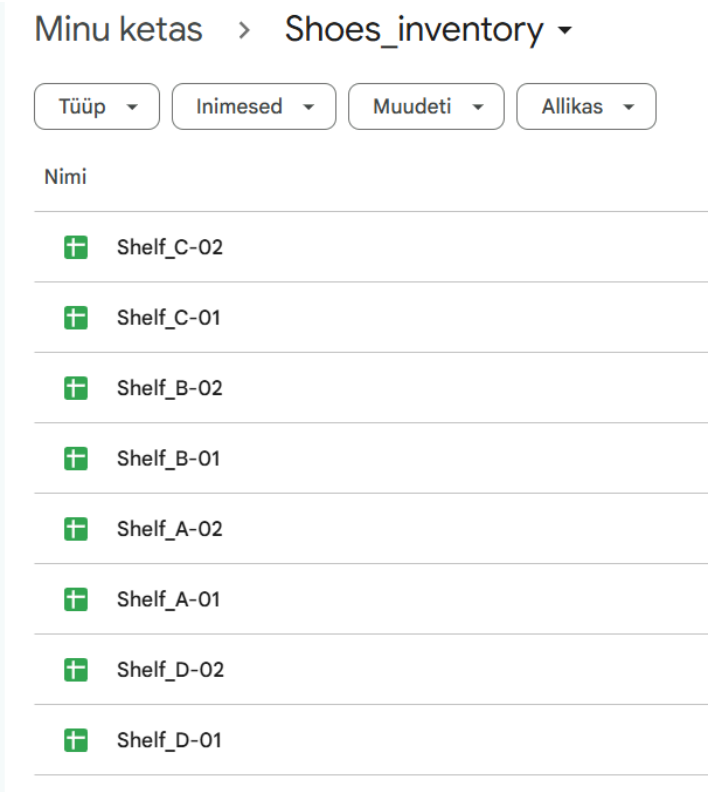
	A	B	C	D
1	product	size	shelf	quantity
2	Reebok Classic	36	D-01	34
3	Reebok Classic	37	D-01	54
4	Reebok Classic	38	D-01	23
5	Reebok Classic	39	D-01	4
6	Reebok Classic	40	D-01	33
7	Reebok Classic	41	D-02	34
8	Reebok Classic	42	D-02	3
9	Reebok Classic	43	D-02	65
10	Reebok Classic	44	D-02	34
11	Reebok Classic	45	D-02	21
12	Adidas Superstа	36	A-01	44
13	Adidas Superstа	37	A-01	66
14	Adidas Superstа	38	A-01	5
15	Adidas Superstа	39	A-01	4
16	Adidas Superstа	40	A-01	66
17	Adidas Superstа	41	A-02	55
18	Adidas Superstа	42	A-02	55
19	Adidas Superstа	43	A-02	6
20	Adidas Superstа	44	A-02	3
21	Adidas Superstа	45	A-02	3
22	Puma Suede	36	B-01	5
23	Puma Suede	37	B-01	33
24	Puma Suede	38	B-01	2
25	Puma Suede	39	B-01	3
26	Puma Suede	40	B-01	66
27	Puma Suede	41	B-02	55
28	Puma Suede	42	B-02	44
29	Puma Suede	43	B-02	33
30	Puma Suede	44	B-02	45
31	Puma Suede	45	B-02	44
32	Nike Air Force 1	36	C-01	34
33	Nike Air Force 1	37	C-01	32
34	Nike Air Force 1	38	C-01	3
35	Nike Air Force 1	39	C-01	4
36	Nike Air Force 1	40	C-01	33
37	Nike Air Force 1	41	C-02	34
38	Nike Air Force 1	42	C-02	2
39	Nike Air Force 1	43	C-02	2
40	Nike Air Force 1	44	C-02	34
41	Nike Air Force 1	45	C-02	33

Function distributeByShelf()

This function reads all data from the Master sheet and automatically separates the rows into new spreadsheets based on the *shelf* value. First, it opens the output folder and removes all old files to ensure that only fresh results are generated. Then it loads the Master sheet, reads every row, and groups the data by shelf code (such as “B-01”, “B-02”, “C-01”, “C-02”, etc.). For each unique shelf, the function creates a new spreadsheet named after that shelf—for example, “Shelf_B-01”. Inside each new spreadsheet, it writes a header row and then adds all products that belong to that shelf, including the product name, size, shelf, and quantity. It also adds an empty “checked” column so that inventory counts can be marked later. In the end, the output folder contains one file per shelf, and each file lists all items stored on that specific shelf.

```
/**************************************
 * TASK 2 – Distribute data by SHELF + add "checked" column
 **************************************/
function distributeByShelf() {

  // OUTPUT folder ID (where shelf-based inventory files will be created)
  let OUTPUT_FOLDER_ID = "1BUrJBVLzFa56qINgDfz3SlN4c1QpQMq1";

  let outputFolder = DriveApp.getFolderById(OUTPUT_FOLDER_ID);

  // Empty the output folder (move old files to trash)
  for (let files = outputFolder.getFiles(); files.hasNext();) {
    files.next().setTrashed(true);
  }

  // Open the same Master sheet
  let masterBook = SpreadsheetApp.openByUrl(
    "https://docs.google.com/spreadsheets/d/1xRMuz8L6UDDV5-S2rLEbRz7x8dxS2GM6ISxtJa_8Fzo/edit"
  );
  let masterSheet = masterBook.getSheetByName("Master");
  let data = masterSheet.getDataRange().getValues();

  // Object to store sheets per SHELF
  let shelfSheets = {};

  // Loop through Master data (skip header row)
  for (let i = 1; i < data.length; i++) {
    let row = data[i];
    let product = row[0];
    let size = row[1];
    let shelf = row[2];
    let quantity = row[3];
```

```
    if (!shelf) {
      continue; // skip rows without shelf
    }

    // Create a new file for this SHELF if not created yet
    if (!shelfSheets[shelf]) {
      let newBook = SpreadsheetApp.create("Shelf_" + shelf);
      let sh = newBook.getSheets()[0];
      sh.clear();

      // Header row – includes quantity and checked
      sh.appendRow(["product", "size", "shelf", "quantity", "checked"]);

      // Move the new file into the output folder
      DriveApp.getFileById(newBook.getId()).moveTo(outputFolder);

      shelfSheets[shelf] = sh;
    }

    // Add row + empty "checked" column for inventory use
    let newRow = [product, size, shelf, quantity, ""];
    shelfSheets[shelf].appendRow(newRow);
  }
}
```



A	B	C	D	E
product	size	shelf	quantity	checked
Nike Air Force 1	41	C-02	34	
Nike Air Force 1	42	C-02	2	
Nike Air Force 1	43	C-02	2	
Nike Air Force 1	44	C-02	34	
Nike Air Force 1	45	C-02	33	

Function runAll()

This function performs both tasks in the correct order—combining all input data and then distributing it—so the user only needs to run one single function to complete the whole workflow.