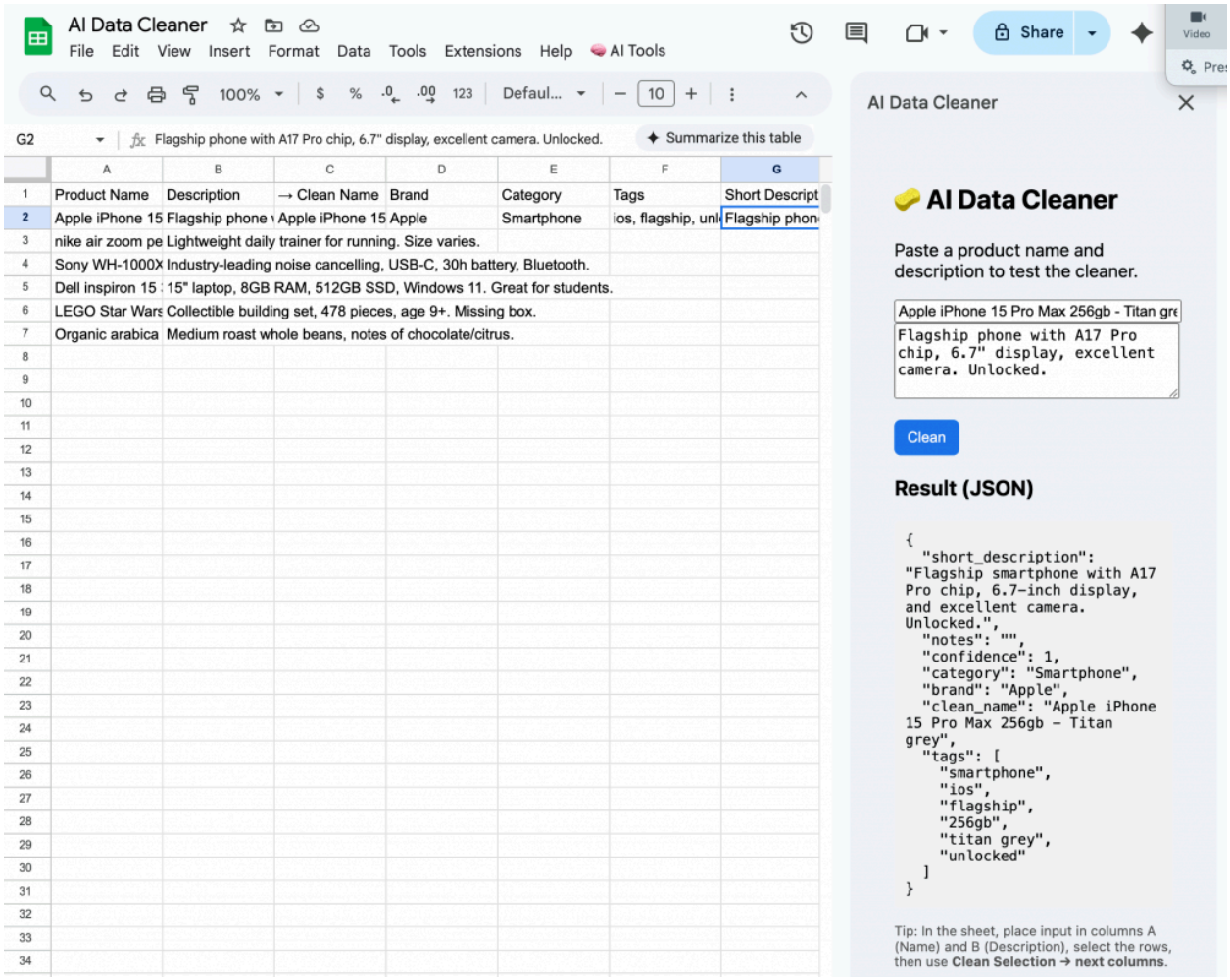
# Exercise 8 — AI Data Cleaner & Categorizer with Google Sheets + Gemini




Managing messy product data is a constant headache — especially when you're working with spreadsheets full of inconsistent titles, missing brands, and vague descriptions.
In this exercise, we'll build an **AI-powered Google Sheets add-on** that automatically cleans, classifies, and enriches product information using the **Gemini API** and **Google Apps Script**.

Source Files
https://github.com/lsvekis/Google-Apps-Script-Gemini-Projects/tree/main/AI%20Data%20Cleaner

By the end, you'll be able to:

✅ Normalize messy product names
✅ Detect brands and categories automatically
✅ Generate short marketing descriptions
✅ Tag items with AI-suggested keywords
✅ Run it directly from your Sheet — no coding knowledge required

---

## 🚀 Overview

We'll connect Google Sheets to the Gemini 2.5 Flash model through a simple Apps Script project.
Once set up, you can select any range of rows and the script will:

1. Send each product name + description to Gemini.
2. Ask for a **JSON response** matching a strict schema.
3. Parse and display clean data in the next columns.

The result is a ready-to-use AI data-cleaning tool built right into Google Sheets.

---

## 🧩 Step 1 — Set up the Script Project

1. Open a Google Sheet.
2. Go to **Extensions** → **Apps Script**.
3. Create three files:

| File name | Purpose |
|---|---|

| | |
|---|---|
| **Code.gs** | The full Apps Script logic |
| **Sidebar.html** | A small UI for testing single products |
| **appsscript.json** | Manifest with required scopes |

Paste in the full code from this project (see below).

4. In **Project Settings** → **Script properties**, add your API key:
    - Name: `GEMINI_API_KEY`
    - Value: *your Gemini API key from [ai.google.dev](ai.google.dev)*
5. Save and run the `onOpen()` function once to authorize the script.

---

## 🧠 Step 2 — How the AI Cleaner Works

The core function is `cleanOneRecord(name, desc)`:

1. **Prompt construction:**
   It tells Gemini exactly which fields to return (clean name, brand, category, tags, short description, confidence, notes).
   It also warns the model to return *"RAW JSON ONLY on a single line"* — no markdown fences.

2. **Gemini request:**
   The function uses `UrlFetchApp.fetch()` to call

   `https://generativelanguage.googleapis.com/v1beta/models/gemini-2.`

`5-flash:generateContent`.

3. **Robust JSON parsing:**
   A helper `safeParseJsonLoose()` cleans up stray ```json fences and safely recovers incomplete objects if Gemini truncates output.

4. **Normalization:**
   The code trims strings, clamps the confidence 0-1, shortens the description ≤ 120 characters, and joins tags into a readable list.

5. **Output:**
   Each cleaned record writes into seven new columns:

   - Clean Name
   - Brand
   - Category
   - Tags
   - Short Description
   - Confidence
   - Notes

---

## ⚙️ Step 3 — Test It with Sample Data

From the new 🧠 **AI Tools** menu in your Sheet:

1. Choose **Insert Sample Data** — this populates demo rows (iPhone, Nike shoes, Sony headphones…).

2. Select columns A and B (Product Name and Description).

3. Choose **Clean Selection** → **next columns**.

4. Watch Gemini fill in clean, organized data instantly.

You can also open **Open Sidebar** and test individual items interactively.

---

## 💡 How the Prompt Keeps Gemini Consistent

The trick is the **strict JSON schema** and explicit rules:

```
{
  "clean_name": "string",
  "brand": "string",
  "category": "string",
  "tags": ["string"],
  "short_description": "string",
  "confidence": "number (0..1)",
  "notes": "string"
}
```

By enforcing this structure and rejecting anything that isn't valid JSON, you get predictable, machine-readable output every time — perfect for automation.

---

## 🔍 Example Result

**Input:**

| Product Name | Description |
| --- | --- |
| Apple iPhone 15 Pro Max 256 GB – Titan Grey | Flagship phone with A17 Pro chip, 6.7" display, excellent camera. Unlocked. |

**Output:**

| Clean Name | Brand | Category | Tags | Short Description | Confidence | Notes |
| --- | --- | --- | --- | --- | --- | --- |
| Apple iPhone 15 Pro Max 256 GB | Apple | Smartphone | smartphone, apple, iphone, flagship, 256gb | Flagship phone with A17 Pro chip and excellent camera. | 0.97 | — |

---

## 🧰 Step 4 — Optional: Add the Custom Function

You can also clean individual cells directly:

```
=AI_CLEAN_ROW(A2,B2)
```

This returns a full JSON string for the given product row.

---

## 📦 Download & Reuse

All ready-to-use files are available in this repo (or ZIP):

- `Code.gs`
- `Sidebar.html`
- `appsscript.json`

→ **Import them into a new Google Sheet's Apps Script editor**
→ Add your Gemini API key
→ You're ready to build your own AI data-cleaning pipeline!

---

## ✨ What You Learned

- How to integrate Gemini with Google Sheets using REST calls
- How to design strict, self-validating JSON prompts
- How to build a sidebar UI and custom functions in Apps Script
- How to safely parse imperfect AI responses

This technique is a foundation for more complex automations — from cleaning e-commerce listings to categorizing support tickets or research data.

# Code.gs

```
/**
* Exercise 8 — AI Data Cleaner & Categorizer (Sheets + Gemini REST)
* Author: Laurence "Lars" Svekis
*
```

```
* Input columns: A=Product Name, B=Description
* Output columns: Clean Name, Brand, Category, Tags, Short Description (≤120), Confidence, Notes
*
* What's included:
* - Menu: Clean Selection → next columns, Clean Sheet…, Open Sidebar, Insert Sample Data
* - Custom function: =AI_CLEAN_ROW(A2,B2)
* - Robust JSON parsing (strips ```json fences, salvages first complete object), strict prompts
*/

////////////////////////
// CONFIG & CONSTANTS //
////////////////////////

const GEMINI_API_KEY = ''; // Prefer Script Properties (Project Settings → Script properties, name: GEMINI_API_KEY)
const GEMINI_MODEL   = 'gemini-2.5-flash';
const BASE_URL       = 'https://generativelanguage.googleapis.com/v1beta';

const DEFAULT_TEMP   = 0.2;
const DEFAULT_TOKENS = 640;
const MERGE_TOKENS   = 768;    // slightly larger budget for retry
const COOLDOWN_MS    = 120;



///////////
// MENU  //
```

```
//////////

function onOpen() {
 SpreadsheetApp.getUi()
   .createMenu('🧠 AI Tools')
   .addItem('Clean Selection → next columns', 'cleanSelectionToRight')
   .addItem('Clean Sheet…', 'cleanSheetPrompt')
   .addItem('Open Sidebar', 'showCleanerSidebar')
   .addSeparator()
   .addItem('Insert Sample Data', 'insertSampleProductData')
   .addToUi();
}

function showCleanerSidebar() {
 const html = HtmlService.createHtmlOutputFromFile('Sidebar').setTitle('AI Data Cleaner');
 SpreadsheetApp.getUi().showSidebar(html);
}


//////////////////////
// KEY & API CLIENT //
//////////////////////

function getApiKey() {
 const p = PropertiesService.getScriptProperties().getProperty('GEMINI_API_KEY');
 return (p && p.trim()) ? p.trim() : (GEMINI_API_KEY || '').trim();
}
```

```
function extractGeminiText(json) {
 try {
  if (json?.error?.message) return `❌ API error: ${json.error.message}`;

  const block = json?.promptFeedback?.blockReason;
  if (block) {
   const details = (json?.promptFeedback?.safetyRatings || [])
    .map(r => `${r.category}:${r.probability}`).join(', ');
   return `❌ Blocked by safety (${block}${details ? ' — ' + details : ''}).`;
  }

  const cand = json?.candidates?.[0];
  if (!cand) return '';

  const parts = cand?.content?.parts || [];
  const text = parts.map(p => p?.text || '').join('').trim();
  if (text) return text;

  const cText = cand?.text;
  if (cText && String(cText).trim()) return String(cText).trim();

  return '';
 } catch (e) { return ''; }
}

function callGeminiJSON(prompt, maxTokens) {
 const apiKey = getApiKey();
```

```
 if (!apiKey) return '❌ Set GEMINI_API_KEY in Script Properties or Code.gs.';

 const payload = {
   contents: [{ role: "user", parts: [{ text: prompt }] }],
   generationConfig: {
     temperature: DEFAULT_TEMP,
     maxOutputTokens: maxTokens || DEFAULT_TOKENS,
     topP: 0.95,
     topK: 40
   },
   // Optional nudge; keep it TOP-LEVEL (not inside generationConfig)
   // responseMimeType: "text/plain"
 };

 const url = `${BASE_URL}/models/${encodeURIComponent(GEMINI_MODEL)}:generateContent?key=${encodeURIComponent(apiKey)}`;
 const options = { method: 'post', contentType: 'application/json', payload: JSON.stringify(payload), muteHttpExceptions: true };

 try {
   const res  = UrlFetchApp.fetch(url, options);
   const code = res.getResponseCode();
   const body = res.getContentText();
   if (code < 200 || code >= 300) return `❌ Error: Gemini HTTP ${code}. ${body}`;
   const text = extractGeminiText(JSON.parse(body));
   return text || '⚠️ No output (see Logs).';
```

```
 } catch (e) { return '❌ Error: ' + e.message; }
}


//////////////////////
// JSON PARSING FIX //
//////////////////////

// Strips ```json fences, then tries to salvage the first complete JSON object.
function safeParseJsonLoose(maybeJson) {
 if (maybeJson == null) return null;
 let s = String(maybeJson).trim();

 // 1) Strip Markdown code fences ```json ... ```
 s = s.replace(/^```(?:json)?\s*/i, '').replace(/```$/i, '').trim();

 // 2) Try direct parse first
 try { return JSON.parse(s); } catch (e) {}

 // 3) Extract the first {...} block and try progressively trimming from the end
 const firstBrace = s.indexOf('{');
 const lastBrace  = s.lastIndexOf('}');
 if (firstBrace === -1 || lastBrace === -1 || lastBrace <= firstBrace) return null;

 for (let end = lastBrace; end >= firstBrace + 1; end--) {
   const candidate = s.slice(firstBrace, end + 1);
```

```
  try { return JSON.parse(candidate); } catch (e) { /* keep trimming */ }
 }
 return null;
}

// Back-compat alias (if referenced elsewhere)
function safeParseJson(maybeJson) {
 return safeParseJsonLoose(maybeJson);
}



////////////////////////////
// PROMPTS & NORMALIZATION //
////////////////////////////

// Strict schema description (for reference)
const CLEAN_SCHEMA = `{
 "clean_name": string,
 "brand": string,
 "category": string,
 "tags": string[],
 "short_description": string,
 "confidence": number,
 "notes": string
}`;

function buildCleanPrompt_(name, desc) {
 const n = (name == null ? '' : String(name)).trim();
 const d = (desc == null ? '' : String(desc)).trim();
```

```
 return [
   'Normalize and categorize this product record.',
   'Return RAW JSON ONLY on a single line. No backticks. No markdown. No prose.',
   'Schema (keys exactly):',
   '{ "clean_name": string, "brand": string, "category": string, "tags": string[], "short_description": string, "confidence": number, "notes": string }',
   'Rules:',
   '- short_description ≤ 120 chars, plain sentence.',
   '- tags: concise, lowercase, 3–6.',
   '- brand: empty string if unknown.',
   '- confidence: 0..1.',
   '',
   'Input record:',
   `Name: ${n || '(blank)'}`,
   `Description: ${d || '(blank)'}`
 ].join('\n');
}

function normalizeObject_(obj) {
 const out = {
   clean_name: String(obj?.clean_name || '').trim(),
   brand: String(obj?.brand || '').trim(),
   category: String(obj?.category || '').trim(),
   tags: Array.isArray(obj?.tags) ? obj.tags.map(t => String(t || '').trim()).filter(Boolean) : [],
   short_description: String(obj?.short_description || '').trim(),
   confidence: Math.max(0, Math.min(1, Number(obj?.confidence) || 0)),
```

```
  notes: String(obj?.notes || '').trim()
 };
 if (out.short_description.length > 120) out.short_description =
out.short_description.slice(0, 117) + '...';
 return out;
}


//////////////////////
// CORE CLEAN FUNCTION //
//////////////////////

function cleanOneRecord(name, desc) {
 const prompt = buildCleanPrompt_(name, desc);
 const raw = callGeminiJSON(prompt, DEFAULT_TOKENS);
 const obj = safeParseJsonLoose(raw);
 if (!obj || typeof obj !== 'object') {
  // Retry with even shorter constraints, still single-line raw JSON only
  const retry = [
   'Return RAW JSON ONLY on a single line. No backticks. No markdown. No
prose.',
   '{ "clean_name": string, "brand": string, "category": string, "tags":
string[], "short_description": string, "confidence": number, "notes": string
}',
   'Constraints: short_description ≤ 120 chars; tags 3–6 lowercase;
confidence 0..1.',
   '',
   'Input:',
   `Name: ${name}`,
```

```
    `Description: ${desc}`
  ].join('\n');

  const raw2 = callGeminiJSON(retry, MERGE_TOKENS);
  const obj2 = safeParseJsonLoose(raw2);
  if (!obj2 || typeof obj2 !== 'object') {
    return { error: '⚠️ Could not parse JSON', raw, raw2 };
  }
  return normalizeObject_(obj2);
 }
 return normalizeObject_(obj);
}


//////////////////////
// SHEETS ACTIONS  //
//////////////////////

function cleanSelectionToRight() {
 const range = SpreadsheetApp.getActiveRange();
 if (!range) { SpreadsheetApp.getUi().alert('Select at least two columns:
Name and Description'); return; }
 const values = range.getValues();
 const out = [];

 for (let i = 0; i < values.length; i++) {
  const name = values[i][0];
  const desc = values[i][1] || '';
  const cleaned = cleanOneRecord(name, desc);
```

```
    if (cleaned.error) {
      out.push([ '(error)', '', '', '', '', '', cleaned.error ]);
    } else {
      out.push([
        cleaned.clean_name,
        cleaned.brand,
        cleaned.category,
        (cleaned.tags || []).join(', '),
        cleaned.short_description,
        cleaned.confidence,
        cleaned.notes
      ]);
    }
    Utilities.sleep(COOLDOWN_MS);
  }

  const sheet = range.getSheet();
  const startRow = range.getRow();
  const nextCol  = range.getColumn() + range.getNumColumns();
  sheet.getRange(startRow, nextCol, out.length, 7).setValues(out);
  SpreadsheetApp.getActive().toast(`Cleaned ${out.length} row(s) →
${colToLetter(nextCol)}:${colToLetter(nextCol+6)}`);
}

function cleanSheetPrompt() {
  const ui = SpreadsheetApp.getUi();
  const res = ui.prompt(
    'Clean Sheet',
```

```
  'Enter input columns (e.g., A,B). Output will start at the next column after your selection.',
  ui.ButtonSet.OK_CANCEL
 );
 if (res.getSelectedButton() !== ui.Button.OK) return;

 const raw = res.getResponseText().trim().toUpperCase().replace(/\s+/g,'');
 const parts = raw.split(',');
 if (parts.length < 1 || !/^[A-Z]+$/.test(parts[0])) { ui.alert('Provide input like A,B (Name,Description)'); return; }
 const colA = letterToCol(parts[0]);
 const colB = parts[1] ? letterToCol(parts[1]) : colA + 1;

 const sheet = SpreadsheetApp.getActiveSheet();
 const lastRow = sheet.getLastRow();
 if (lastRow < 2) { ui.alert('No data to clean.'); return; }

 const range = sheet.getRange(2, Math.min(colA,colB), lastRow-1, 2);
 range.activate();
 cleanSelectionToRight();
}


//////////////////////
// CUSTOM FUNCTION  //
//////////////////////

// Use: =AI_CLEAN_ROW(A2, B2)
function AI_CLEAN_ROW(name, desc) {
```

```
 return JSON.stringify(cleanOneRecord(name, desc));
}


//////////////////
// SAMPLE DATA    //
//////////////////

function insertSampleProductData() {
 // 9 columns: Name, Description, → Clean Name, Brand, Category, Tags, Short Description, Confidence, Notes
 const headers = ['Product Name','Description','→ Clean Name','Brand','Category','Tags','Short Description','Confidence','Notes'];
 const rows = [
   ['Apple iPhone 15 Pro Max 256gb - Titan grey!','Flagship phone with A17 Pro chip, 6.7" display, excellent camera. Unlocked.'],
   ['nike air zoom pegasus 39 Mens','Lightweight daily trainer for running. Size varies.'],
   ['Sony WH-1000XM5 ANC Headphones','Industry-leading noise cancelling, USB-C, 30h battery, Bluetooth.'],
   ['Dell inspiron 15 3525 laptop Ryzen 5','15" laptop, 8GB RAM, 512GB SSD, Windows 11. Great for students.'],
   ['LEGO Star Wars set 75300','Collectible building set, 478 pieces, age 9+. Missing box.'],
   ['Organic arabica coffee beans 1kg','Medium roast whole beans, notes of chocolate/citrus.']
 ];

 const WIDTH = 9;
```

```
 const paddedRows = rows.map(r => {
   const row = r.slice(); // [Name, Description]
   while (row.length < WIDTH) row.push('');
   return row;
 });

 const data = [headers, ...paddedRows];

 const sheet = SpreadsheetApp.getActiveSheet();
 sheet.clearContents();
 sheet.getRange(1, 1, data.length, WIDTH).setValues(data);

 SpreadsheetApp.getActive().toast('✅ Sample product data inserted (A–I).
Select A:B and run “Clean Selection → next columns”.');
}



//////////////
// HELPERS  //
//////////////

function letterToCol(letter) {
 let col = 0;
 for (let i=0; i<letter.length; i++) col = col*26 + (letter.charCodeAt(i)-64);
 return col;
}

function colToLetter(col) {
 let temp='', letter='';
```

```
 while (col>0) {
   temp=(col-1)%26;
   letter=String.fromCharCode(temp+65)+letter;
   col=(col-temp-1)/26;
 }
 return letter;
}
```

## Sidebar.html

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="utf-8">
   <style>
     body { font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial, sans-serif; padding: 16px; }
     input, textarea { width: 100%; }
     button { background: #1a73e8; color: #fff; border: 0; padding: 8px 12px; border-radius: 6px; cursor: pointer; }
     pre { background: #f1f3f4; padding: 10px; border-radius: 6px; white-space: pre-wrap; }
     .small { color:#555; font-size:12px; }
     code { background:#eef; padding:2px 4px; border-radius:4px; }
   </style>
 </head>
 <body>
   <h2>🧽 AI Data Cleaner</h2>
```

```
  <p>Paste a product name and description to test the cleaner.</p>

  <input id="name" placeholder="Product name" />
  <textarea id="desc" rows="4" placeholder="Description
(optional)"></textarea>
  <p><button id="run" onclick="run()">Clean</button></p>

  <h3>Result (JSON)</h3>
  <pre id="out">Waiting…</pre>
  <p class="small">Tip: In the sheet, place input in columns A (Name) and
B (Description), select the rows, then use <b>Clean Selection → next
columns</b>.</p>

  <script>
   function run(){
    const btn = document.getElementById('run');
    const name = document.getElementById('name').value || '';
    const desc = document.getElementById('desc').value || '';
    document.getElementById('out').textContent = '⏳ Cleaning…';
    btn.disabled = true;
    google.script.run
     .withSuccessHandler(r => {
      document.getElementById('out').textContent = JSON.stringify(r, null,
2);
      btn.disabled = false;
     })
     .withFailureHandler(e => {
      document.getElementById('out').textContent = 'Error: ' +
e.message;
```

```
          btn.disabled = false;
        })
        .cleanOneRecord(name, desc);
    }
  </script>
 </body>
</html>
```